UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS FÍSICAS

Departamento de Informática y Automática

TESIS DOCTORAL

Procesador concurrente para bases de datos

MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

José Jaime Ruz Ortiz

DIRECTOR:

Emilio Luque Fadón

Madrid, 2015

José Jaime Ruz Ortiz

PROCESADOR CONCURRENTE PARA BASES DE DATOS

Departamento de Informática y Automática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid
1982

Edita e imprime la Editorial de la Universidad
Complutense de Madrid. Servicio de Reprografía
Noviciado, 3 Madrid-8
Madrid, 1981
Xerox 9200 XB 480
Depósito Legal: M-36683-1981

Autor: JOSE J. RUZ ORTIZ

PROCESADOR CONCURRENTE PARA BASES DE DATOS

Director: Emilio Luque Fadón
Doctor en Ciencias Físicas, Prof. Agregado de
Física Industrial (Automática)

UNIVERSIDAD COMPLUTENSE DE MADRID
Facultad de Ciencias Físicas
Departamento de Informática y
Automática

Año 1980

A mis padres

Deseo expresar mi agradecimiento al Prof. Dr. D. Emilio Luque Fadón por el entusiasmo y dedicación con que ha dirigido el presente trabajo.

A Dª. Ana Ripoll Aracil por la inestimable ayuda prestada en la fase preliminar de estudio y recopilación de información, sin duda la más crítica y decisiva de todas.

A D. Alfredo Bautista Paloma por las valiosas sugerencias a lo largo de todo el trabajo.

A la Sta. Rosa Fernández de las Heras que llevó a cabo la fase de realización física.

A D. Román Hermida Correa cuyas críticas y valiosas estimaciones han aclarado buen número de las ideas que aquí exponemos.

Al Prof. Dr. D. Francisco Tirado Fernández por su constante colaboración.

A todos los compañeros del Departamento y muy especialmente a su Director, el Prof. Dr. D. Mariano Mellado Rodríguez, que desde su incorporación al mismo nos ha brindado todo su apoyo para llevar a buen término la presente Memoria.

Finalmente, a Dª. Margarita Pérez-Mosso por el esmero mostrado en el mecanografíado de la memoria.

I N D I C E

pág.

pág.

PROPOSITO Y DESARROLLO DE LA PRESENTE MEMORIA

El trabajo que exponemos en la presente memoria tiene como objetivo fundamental la concepción y diseño de un Procesador para gestión de Bases de Datos que supere las limitaciones que los ordenadores convencionales imponen a este tipo de aplicaciones.

Para ello, teniendo siempre presente las actuales posibilidades de la tecnología hardware, las diferentes fases de desarrollo giran en torno a la consecución de dos objetivos:

a) Identificación e implementación directa sobre el Procesador de las funciones propias del almacenamiento y recuperación de información.

b) Adaptación de su organización interna a la naturaleza multiusuario de una Base de Datos.

Las distintas etapas de investigación y desarrollo las exponemos divididas en cuatro capítulos.

En el primero, tras describir brevemente los objetivos que se persiguen con el establecimiento de una Base de Datos y estudiar comparativamente los diferentes modelos lógicos utilizados en la actualidad, se ponen de manifiesto las ventajas del modelo relacional. Por la claridad conceptual e independencia de los factores de implementación con que esta alternativa plantea la problemática de la modelación y acceso a una Base de Datos, ampliamente expuestas en el capítulo, el modelo relacional se escoge como forma de representación interna de datos para el Procesador.

En el capítulo segundo, se exponen los principios básicos que inspiran la organización general del Procesador propuesto; principios

que vienen a superar las principales causas estructurales del bajo rendimiento de los ordenadores convencionales. Se estudia su arquitectura general, haciendo especial énfasis en el carácter completo de su repertorio de instrucciones. Se describe el comportamiento funcional de las células integrantes a fin de poner de manifiesto la ejecución concurrente de los accesos. Finalmente, se esbozan las líneas generales de las políticas de concurrencia utilizables.

El capítulo tercero recoge todo lo concerniente a la fase de implementación física. En él se describen, de forma exahustiva, los mecanismos específicos de las diferentes unidades de las células componentes del Procesador. Especial atención se presta al dispositivo de evaluación de cualificaciones, responsable directo de la capacidad selectiva de datos del Procesador.

Finalmente, en el cuarto capítulo se hace un estudio comparativo de rendimiento del Procesador propuesto frente a un sistema convencional y frente a otro procesador, específicamente diseñado para gestión de Bases de Datos. Se obtienen fórmulas analíticas de los tiempos promedios de respuesta respectivos para diferentes tipos de accesos, y se muestran en forma gráfica los resultados obtenidos.

M E M O R I A

C A P I T U L O I

BASES DE DATOS

I.1.- CONCEPTO GENERAL DE BASE DE DATOS

La adopción de decisiones respecto a los fenómenos, cada vez más complejos, del mundo real, demanda la disponibilidad de grandes cantidades de información relativa a los factores determinantes de la evolución de tales fenómenos. La información representa la interpretación de los datos obtenidos por observación y medida de la realidad. Los mecanismos de interpretación de datos pueden ser de muy variada naturaleza, pero, la validez de la información resultante, en cuanto a su utilidad para la toma de una mejor decisión, será tanto mayor cuanto más fiables sean los sistemas de obtención de datos y menor el tiempo transcurrido entre la generación de éstos y su disponibilidad en los centros de decisión.

Entendiendo por sistema real aquella parte del mundo físico que se considera de interés para la elaboración de decisiones, una Base de Datos (BD) constituye, en un sentido amplio, una representación abstracta y esquemática de un sistema real que, facilitando la disponibilidad de sus datos asociados, ayuda a un conjunto de usuarios en la adopción de decisiones. La utilidad de una BD descansa, pues, en el hecho de que las cuestiones relativas a ciertos aspectos de un sistema real, pueden responderse consultando los datos de la base que representan dichos aspectos. Para que en todo momento la representación sea fiel, una BD debe reflejar todos los cambios habidos en el sistema real, es decir, debe ser factible su actua-

lización. Consulta y actualización serán, pues, las dos formas de interacción entre una BD y su entorno. En la figura I-1 se resumen, en forma gráfica, estas interacciones.

La representación del sistema real en la BD se lleva a efecto por medio de un modelo de datos. Se trata, no de un modelo de comportamiento, pues la dimensión tiempo no forma parte de su definición, sino de una descripción estática de los objetos y relaciones que componen el sistema y cuyo conocimiento resulta de interés para la inferencia de información por los usuarios de la base.

Las funciones de actualización y consulta se realizan por medio de un lenguaje de manipulación asociado al modelo de datos.

Modelo de datos y lenguaje de manipulación asociado definirán la estructura conceptual y el comportamiento funcional de una BD.

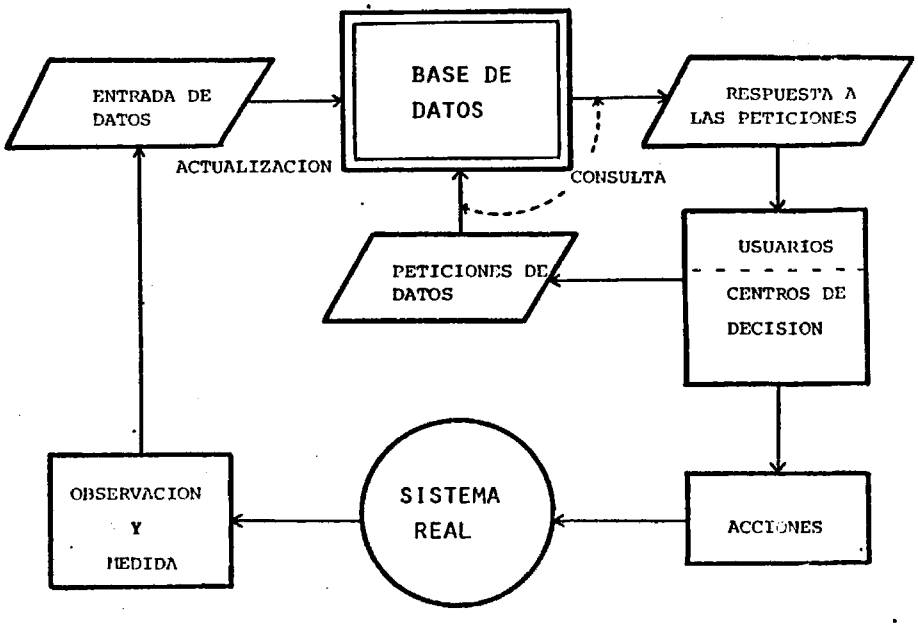


Fig. I-1. Relaciones entre una BD y su entorno

I-2.- EVOLUCION DE LA GESTION DE DATOS: BASE DE DATOS COMPUTARIZADA

Las actuales Bases de Datos computarizadas son el resultado de una serie de esfuerzos tendentes a facilitar la gestión automatizada de grandes cantidades de información. Su origen podemos situarlo en los primitivos métodos manuales, y su evolución ha sido propiciada por las innovaciones de la tecnología informática (hardware y software). La disponibilidad de recursos tecnológicos, cada vez más potentes, ha hecho posible la concepción de sistemas cada vez más complejos en el campo de la gestión de información. A veces, los nuevos conceptos han estado fuertemente influenciados por la metodología impuesta por la filosofía de funcionamiento del ordenador digital. Este hecho ha dificultado en muchos casos una definición precisa de los objetivos finales que se perseguían con el establecimiento de una base de datos en la que no apareciesen detalles de implementación. No obstante, los recientes esfuerzos en el estudio de la semántica de Base de Datos, especialmente a raíz de la aparición del modelo relacional de Codd, permiten, en la actualidad, una formulación conceptualmente clara del problema.

Los primeros sistemas mecanizados de gestión de datos que podemos considerar son los que se basaban en la tecnología de programa cableado. Se trataba de clasificadoras, generadores de informes, etc., que utilizaban fichas perforadas como soporte de datos. Aunque esta tecnología proporcionaba considerables ventajas frente a los procedimientos manuales, la rigidez del procesamiento, baja velocidad impuesta por la naturaleza mecánica de los dispositivos y limitación de la capacidad de almacenamiento, hacía difícil su evolución.

Con la aparición del ordenador de programa almacenado, utilizando dispositivos de cinta magnética como medio de almacenamiento masivo de

datos, la gestión de información experimentó una mejora considerable. La velocidad se incrementó de 4 a 5 órdenes de magnitud; la capacidad de almacenamiento aumentó a igualdad de volumen físico; y la complejidad del procesamiento creció como consecuencia de la flexibilidad introducida por el programa almacenado. A pesar de todo, el cambio fue más de tipo cuantitativo que cualitativo ya que, como en la anterior etapa, el procesamiento continuó siendo secuencial. Un detalle de esta similitud cualitativa lo constituye la utilización de idéntica terminología: archivo, registro y campo son términos surgidos en la tecnología de fichas que se siguen utilizando con la misma significación.

La primera orientación del ordenador a la gestión de datos tuvo lugar en el campo del software al independizarse la descripción de los archivos del programa de tratamiento. En la etapa anterior, la descripción iba implícita en la lógica del programa, lo que hacía extremadamente complejo el procesamiento de un archivo por diferentes programas. Los Sistemas de Gestión de Ficheros permitieron la separación entre descripción de datos y descripción de procesamiento, facilitando la compartición de archivos por distintos programas. La DATA DIVISION de COBOL pone de manifiesto esta característica.

La aparición de los Dispositivos de Almacenamiento de Acceso Directo (DAAD), en forma de grandes discos y tambores magnéticos, capaces de almacenar cientos de millones de caracteres, produjo un cambio cualitativo en la metodología del procesamiento y potencia de los sistemas de gestión. En un DAAD cualquier registro puede ser accedido en menos de un segundo. Evidentemente, esto no supone mayor velocidad que la conseguida en el acceso secuencial a registros adyacentes, pero la posibilidad de acceder en forma aleatoria, reduce el número de accesos necesarios en la mayoría de

las aplicaciones y posibilita la consulta "on-line" de los registros.

Un obstáculo que hubo que superar en los DAADs fue el relativo a la direccionabilidad por contenido. Al ser dispositivos de acceso por dirección, la accesibilidad por contenido obliga a disponer de un método que, a partir de dicho contenido, obtenga la dirección física del registro. La solución de este problema dio origen a las técnicas de organización de archivos. Básicamente consisten en la utilización de dos tipos de información: la propiamente dicha del archivo, distribuida según un orden, y una información índice o estructural conteniendo la forma de distribución de la información principal. El acceso a un registro determinado va precedido por el procesamiento de la información índice.

El término Base de Datos (BD) aparece para designar a los sistemas con control centralizado de datos surgidos del proceso de integración de archivos independientes. La adopción de este control confiere las siguientes características:

- Reducción de la redundancia de datos.- En los sistemas anteriores al advenimiento de las BDs, cada aplicación mantenía sus propios archivos particulares, cuando, con frecuencia, más de una aplicación necesitaba datos comunes. Ocurría, pues, que los mismos datos aparecían duplicados en archivos diferentes con el consiguiente desaprovechamiento de memoria. En una BD, al integrarse los archivos de diferentes aplicaciones, los datos se almacenan una sola vez.

- Eliminación de datos inconsistentes.- Se trata de una consecuencia inmediata de la anterior. En efecto, al no existir datos redundantes, se elimina la posibilidad de inconsistencia entre ellos, esto es, que los mismos datos aparezcan con diferentes valores en distintos archivos, por el hecho de no actualizarse simultáneamente.

- Posibilidad de la compartición de datos.- Aunque esta característica es una consecuencia directa del proceso de integración, las nuevas aplicaciones de la base han de poder operar sin introducir datos nuevos, si los necesitados por éstas están ya contenidos en aquélla. No obstante, para que la compartición sea eficiente, una BD debe garantizar la independencia de sus datos en un doble aspecto:

a) independencia física que garantiza la posibilidad de introducir modificaciones en los soportes físicos y métodos de acceso sin modificar la interface externa y, por tanto, los programas de aplicación existentes. La independencia física exige la diferenciación de dos niveles en una BD: 1) nivel físico, que contempla la forma de almacenamiento de datos sobre los dispositivos de memoria y 2) nivel lógico, que recoge la forma con que los datos se presentan al usuario.

b) independencia lógica que garantiza la posibilidad de introducir nuevos datos y lazos semánticos a nivel lógico (con su repercusión a nivel físico) sin afectar a los ya existentes y, por tanto, a los programas de aplicación. Surge, pues, la necesidad de diferenciar dos subniveles dentro del nivel lógico: 1) nivel externo o subesquema, particular para cada aplicación o conjunto de aplicaciones afines y 2) nivel general o esquema, que contempla la totalidad de datos y lazos semánticos de la base.

- Control de la seguridad de los datos.- Característica muy importante en determinadas aplicaciones que necesitan proteger sus datos contra usos desautorizados.

- Mantenimiento de la integridad.- Asegura el cumplimiento de determinadas restricciones que garantizan la precisión semántica de los datos.

El conjunto de programas y estructuras de datos que en una BD mantiene la correspondencia entre los diferentes niveles, así como las restricciones de integridad y seguridad y otras características establecidas, recibe el nombre de Sistema de Gestión de Base de Datos (SGBD). Desde un punto de vista funcional el SGBD proporciona las herramientas para la descripción de datos en cada uno de los niveles de la base.

Considerando los tres niveles anteriormente definidos, una BD computarizada tendría los componentes mostrados en la figura I-2([1]).

Los usuarios acceden a la base desde un programa codificado con un lenguaje de programación (P-I) que podrá ser convencional o específico de interrogación (querry language). En cualquier caso, se denomina sublenguaje de datos al subconjunto que resulta de eliminar los recursos de proceso del lenguaje utilizado. Es decir, un sublenguaje de datos sólo considera aquellos aspectos relacionados directamente con los accesos a la base (recuperación y almacenamiento). Cada usuario dispone de un área de trabajo (AT-I) para realizar sus interacciones con la base (transmisión y recepción). Los accesos hacen referencia al subesquema particular de cada usuario, esto es, al submodelo de datos extraído del esquema y definido con el correspondiente lenguaje de definición del subesquema. La correspondencia subesquema-esquema es mantenido por el SGBD.

El esquema representa la información total contenida en la base y definida, según un modelo lógico, por medio de un lenguaje de definición del esquema. La correspondencia entre esquema y estructura física de almacenamiento -que habrá de tener en cuenta la forma de distribución de datos sobre los soportes de memoria así como las estrategias de acceso a los mismos- es mantenida por el SGBD.

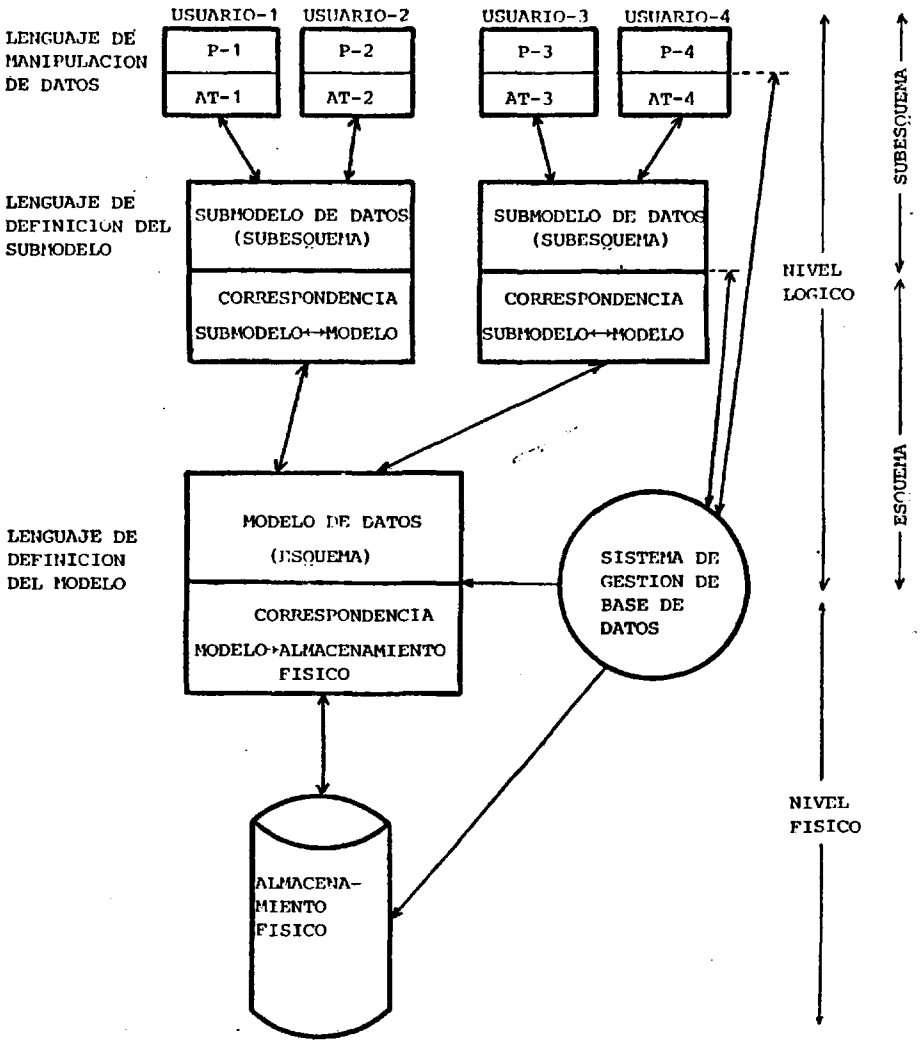


Figura I-2.- Componentes de una BD computatizada

Como pondremos de manifiesto en el capítulo II, muchos de los conceptos ligados al estudio de una BD derivan directamente de las necesidades impuestas por la filosofía de funcionamiento del ordenador convencional. Teniendo en cuenta el objetivo del presente trabajo y la prolijidad de tales conceptos, tan sólo consideraremos, en lo que resta del presente capítulo, los aspectos de más alto nivel relativos a una BD, esto es, los que hacen referencia a sus posibilidades de utilización y que, en definitiva, definen su estructura conceptual y su comportamiento funcional.

I-3.- MODELOS LOGICOS DE DATOS Y SUBLENGUAJES DE MANIPULACION ASOCIADOS

Estudiaremos en este apartado los tres modelos lógicos utilizados en la actualidad para representar un sistema real sobre una BD, así como los sublenguajes de manipulación que permiten su consulta y actualización.

Comenzaremos definiendo, de una manera formal, los elementos del sistema real que se contemplan en cualquier modelo lógico. A continuación, describiremos la forma particular que estos elementos adoptan en cada uno de los modelos: jerárquico, red y relacional.

Definiremos una entidad (e_i) como "algo" que existe y puede identificarse unívocamente; una persona o un suceso son ejemplos de entidades. Mediante predicados podemos agrupar las diferentes entidades -de interés en el sistema real- en conjuntos o tipos de entidades (E_i), tales como "empleados", "departamentos", "proyectos", etc. Estos conjuntos no tienen por qué ser mutuamente disjuntos, por ejemplo, una entidad que pertenece al conjunto de entidades "jefes", también pertenece al conjunto de entidades "empleados" (ya que "jefes" es un subconjunto de "empleados").

Las entidades de un sistema real pueden estar relacionadas a través de conexiones, por ejemplo, "padre-hijo" es una conexión entre dos entidades "personas". Formalmente, dados n conjuntos de entidades E_i (no necesariamente distintos) una conexión (c_i) es un n-tuple ordenado de la forma $(e_1, e_2, \ldots, e_n)$ $\quad e_i \in E_i$ $(i = 1, n)$. Es decir, una conexión es un elemento del producto cartesiano $E_1 \times E_2 \times \ldots E_n$. A cualquier subconjunto $C \subseteq E_1 \times E_2 \times \ldots E_n$ se le denomina conjunto de conexiones o lazo semántico. Es importante hacer notar que los n conjuntos E_i no tienen por qué ser diferentes; un "matrimonio" es un lazo semántico entre dos entidades del mismos conjunto "personas".

Cuando en la definición de una conexión intervienen conjuntos de entidades repetidos, es necesario diferenciar la función desempeñada por cada uno de ellos. Esto se hace asociando un rol diferente a cada entidad.

Por ejemplo, "marido" y "mujer" son los roles respectivos de dos entidades "personas" en la conexión "matrimonio". El orden de las entidades, en la definición de conexión, puede definir de forma implícita los roles de las diferentes entidades. Si el orden se omite, los roles hay que explicitarlos.

Atendiendo al tipo de correspondencia que un lazo semántico define entre dos conjuntos de entidades E_1 y E_2 distinguiremos(2):

- lazo semántico del tipo (1:1), una entidad de E_1 está conectada con una sola entidad de E_2 y viceversa: $E_1 \xrightarrow{1:1} E_2$
- lazo semántico del tipo (1:N), una entidad de E_1 está conectada a varias entidades de E_2: $E_1 \xrightarrow{1:N} E_2$
- lazo semántico del tipo (N:1), varias entidades de E_1 están conectadas a una sola entidad de E_2: $E_1 \xrightarrow{N:1} E_2$

- lazo semántico del tipo (N:M), varias entidades de E_1 están conectadas a varias entidades de E_2: $E_1 \xrightarrow{N:M} E_2$

Así, el lazo semántico "matrimonio" entre los dos tipos de entidades "hombre" y "mujer" podría ser, según todas las posibles formas de concebir un matrimonio, de los siguientes tipos:

hombre $\xrightarrow{1:1}$ mujer (matrimonio convencional

hombre $\xrightarrow{1:N}$ mujer (poligamia)

hombre $\xrightarrow{N:1}$ mujer (poliandria)

hombre $\xrightarrow{N:M}$ mujer (matrimonio grupal)

La información acerca de una entidad o una conexión se obtiene por observación o medida y se expresa por un conjunto de pares (atributo, valor). Así, "3", "rojo" y "Pedro" son posibles valores de posibles atributos de entidades. Los valores se clasifican en conjuntos de valores (dominios) (V_i) por medio de predicados asociados; "medida","color" y "nombre" son conjuntos de valores. Un atributo (A) puede definirse formalmente como una aplicación de un conjunto de entidades o un conjunto de conexiones sobre un conjunto de valores: $E_i \xrightarrow{A} V_i$ o $C_i \xrightarrow{A} V_i$; los atributos manifiestan propiedades de las entidades o conexiones de interés en el contexto de utilización de la base.

La decisión de considerar un objeto determinado del sistema real como entidad o conexión en una base de datos, es una tarea del diseñador lógico de la base y no está sujeta a reglas fijas. Así, un "vuelo" en una compañía aérea se podría considerar como una entidad con sus atributos correspondientes (avión, ciudad de partida, ciudad de llegada, piloto, etc.) o como una conexión entre los conjuntos de entidades "aviones" y "pilotos".

Definiremos, a continuación, un ejemplo de sistema real que representaremos, en los subapartados siguientes, según los tres modelos lógicos más utilizados en la actualidad y citados al comienzo del apartado.

Se trata de las actividades de una compañía aérea en la que se contemplan tres tipos de objetos: aviones, pilotos y vuelos. De cada objeto se consideran de interés para la base las siguientes características:

AVION: nº (AV#), nombre (AVNOM), capacidad (CAP) y ciudad base (CB)

PILOTO: nº (PI#), nombre (PINOM), ciudad de residencia (CR)

VUELO: nº (VU#), piloto (PI#), avión (AV#), ciudad de origen (CO), ciudad de destino (CD), hora de partida (HP) y hora de llegada (HL).

Entre los tres objetos existen los siguientes lazos semánticos:

(1) un piloto puede conducir cualquier avión: PILOTO $\xrightarrow{1:N}$ AVION

(2) un piloto puede cubrir cualquier vuelo: PILOTO $\xrightarrow{1:N}$ VUELO

(3) un avión puede ser conducido por cualquier piloto: AVION $\xrightarrow{1:N}$ PILOTO

(4) un avión puede cubrir cualquier vuelo: AVION $\xrightarrow{1:N}$ VUELO

Los lazos semánticos (1) y (3) son equivalentes a:

(5) cualquier piloto puede conducir cualquier avión: PILOTO $\xrightarrow{N:M}$ AVION

I.3.1.- Modelo jerárquico

Por razones históricas, el modelo jerárquico es muy popular. Su utilización está bastante extendida en los sistemas actuales de BD, siendo IMS (Information Management System) de IBM ([3]) su más caracterizado representante. Surgió por evolución de las estructuras de datos que se utilizaban cuando la mayor parte del procesamiento de datos se realizaba sobre

dispositivos de almacenamiento secuencial y existía una mínima distinción entre modelo lógico de datos y estructura física de almacenamiento.

Adoptando la terminología que IBM utiliza en IMS, las entidades se denominan en este modelo segmentos, los conjuntos de entidades tipos de segmentos y los atributos campos. La conexiones permitidas entre segmentos son de tipo árbol o jerárquicas, como se muestra en la figura I-3.

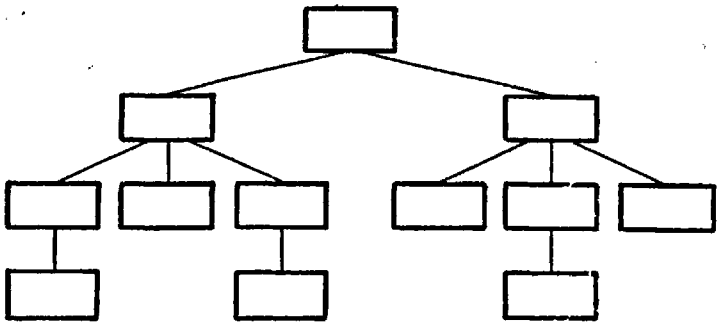


Figura I-3.- Estructura jerárquica

Cada nodo del árbol es un segmento. El nivel más alto de la jerarquía tiene un solo nodo que se llama raíz. Con excepción de la raíz, todo nodo está vinculado a otro de nivel más alto que se denomina predecesor inmediato. Ningún segmento puede tener más de un predecesor inmediato. En cambio, todo segmento puede tener uno o más segmentos conectados en un nivel más bajo y denominados sucesores inmediatos.

Como puede observarse, el modelo jerárquico sólo permite la representación directa de lazos semánticos tipo (1:N). Los de tipo (N:M) han de expresarse en términos de aquéllos.. Así, para la representación del sistema real propuesto en el apartado anterior, según el modelo jerárquico, podemos considerar dos tipos de entidades: piloto y avión-vuelo, y definir consecuentemente dos tipos de segmentos: PILOTO y AVION-VUELO con sus res-

pectivos campos. De esta forma, los lazos semánticos pueden representarse con un árbol de dos niveles (figura I-4).

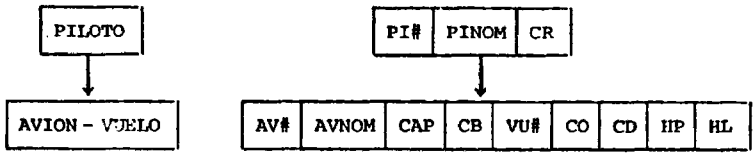


Figura I-4

El número de jerarquías de este tipo (ocurrencias) o lo que es lo mismo, el número de segmentos raíz será igual al número de pilotos. En la figura I-5 se muestran los correspondientes a los pilotos n° 1 y n° 2. Cada segmento piloto está conectado a tantos segmentos avión-vuelo como vuelos diferentes realice dicho piloto.

1	JUAN	MADRID							
2	B707	200	MADRID	IB100	MADRID	SEVILLA	11	12
8	B747	350	NEW YORK	IB110	MADRID	NEW YORK	10	18
4	B707	200	MADRID	IB120	MADRID	PARIS	10	12

2	PEDRO	SEVILLA							
2	B707	200	MADRID	IB101	SEVILLA	MADRID	22	23
4	CARA	200	MADRID	IB109	MADRID	ROMA	16	19

Figura I-5

El sublenguaje de datos asociado a este modelo se compone de un conjunto de sentencias que permiten recorrer las estructuras jerárquicas

para acceder al dato deseado. La unidad de acceso, es decir, la mínima cantidad de información que puede transferirse por una sentencia es un segmento. Las sentencias se utilizan desde el lenguaje de programación en el cual se codifica el proceso que accede a la base. En IMS de IBM, el sublenguaje de datos se denomina DL/I ([4]) y puede utilizarse desde los lenguajes PL/I, COBOL o ASSEMBLER del S/360. Las sentencias DL/I más importantes son las siguientes:

- GU (GET UNIQUE). Recupera directamente un segmento
- GN (GET NEXT). Recupera el segmento siguiente en secuencia
- GNP (GET NEXT WITHIN PARENT). Recupera el segmento siguiente en secuencia bajo el predecesor inmediato actual
- GHU, GHN, GHNP (GET HOLD). Equivalentes a las tres anteriores cuando la siguiente sentencia es DLET o REPL
- DLET (DELETE). Borrra un segmento al que se llega por alguna sentencia GET HOLD
- REPL (REPLACE). Actualiza un segmento al que se llega por alguna sentencia GET HOLD
- ISRT (INSERT). Inserta un nuevo segmento

Cada sentencia, en general, va acompañada de dos tipos de parámetros: la dirección del área de E/S a través de la cual se realiza el intercambio y una o más condiciones de cualificación (cc). Una cc consta de un nombre de segmento opcionalmente seguido por una condición. Si la condición se omite cualquier ocurrencia del segmento indicado satisface la cc. Una condición consta de un conjunto de expresiones de comparación conectados por los operadores booleanos "Y" y "O". Cada expresión de comparación es un triplete <campo, operador de comparación, valor> donde el campo debe

pertenecer al segmento especificado y el operador de comparación puede ser uno cualquiera del conjunto $\{=, \neq, <, \leq, \geq, >\}$. Todas las operaciones de comparación se realizan bit a bit de izquierda a derecha. Las sentencias GU y ISRT necesitan especificar, además de la CC, el camino jerárquico completo desde la raíz. GN y GNP no necesitan especificar ningún camino jerárquico, pero, en caso de hacerlo, éste puede comenzar en cualquier nivel y no necesariamente en la raíz.

La principal ventaja del modelo jerárquico aparece cuando las conexiones a representar son por naturaleza jerárquicas. Sin embargo, en la mayoría de los casos esto no ocurre. En general, aparecen conexiones del tipo (N:M) que, al expresarlas implícitamente con la repetición de datos, como hicimos con el ejemplo propuesto, ocasionan anomalías de almacenamiento. Además, las estructuras jerárquicas presentan distinto grado de dificultad a la expresión de preguntas análogas. Así, consideremos los procedimientos DL/I necesarios para responder a las dos preguntas siguientes:

P1 ¿Cuáles son los aviones conducidos por el piloto nº 2?

```
     GU PILOTO (P1 ≠ '2')
P    GNP AVION-VUELO
  <SEGMENTO ENCONTRADO> --NO--> [FIN]
     IMPRIMIR AV≠
     GO TO P
```

P2: ¿Cuáles son los pilotos que conducen el avión nº 4

```
     GU PILOTO (al comienzo de la
                jerarquía)
P    GN PILOTO
  <SEGMENTO ENCONTRADO> --NO--> [FIN]
     GNP AVION-VUELO (AV≠ = '4')
  <SEGMENTO ENCONTRADO> --NO--> GO TO P
     IMPRIMIR PI≠
     GO TO P
```

Se puede observar que, aunque las preguntas P1 y P2 son simétricas en el sentido de ser una la inversa de la otra, los procesimientos DL/I son asimétricos. Ello pone de manifiesto una de las principales desventajas del modelo jerárquico: una complejidad innecesaria. El usuario se ve obligado a invertir tiempo y esfuerzo para resolver problemas propios del modelo y extrínsecos a la naturaleza de las preguntas formuladas.

Para poner de manifiesto las anomalías de almacenamiento presentadas por el modelo jerárquico, consideremos las tres operaciones típicas de modificación de una base:

- Inserción.- No es posible insertar un segmento sin que exista su predecesor inmediato. Así, refiriéndonos al ejemplo, no sería posible almacenar la información referente a un vuelo, si previamente no se ha fijado el piloto que lo cubrirá.

- Borrado.- Si se necesita eliminar un segmento, se eliminan con él todos los sucesores inmediatos. Así, si el piloto n° 1 deja de formar parte de la compañía y se elimina de la base, desaparecerá toda la información referente a los vuelos que realizaba.

- Actualización.- Si se necesita actualizar algún campo de un segmento s1, sucesor de otro s2; existiendo entre sus respectivas entidades un lazo semántico del tipo $E_1 \xrightarrow{1:N} E_2$, habría que recorrer la estructura completa a la búsqueda de todos los segmentos s1, si se quiere evitar el riesgo de inconsistencia. Así, si la ciudad base del avión n° 2 se cambia de Madrid a Sevilla, habría que actualizar todos los segmentos avión-vuelo donde aparezca dicho avión.

I.3.2.- Modelo Red

Este modelo viene tipificado por el sistema propuesto por el

Data Base Task Group (DBTG) de CODASYL ([5])([6]). Las entidades se representan mediante registros, los atributos por ítems de datos y los conjuntos de entidades por tipos de registros. Para la definición de conexiones se utilizan estructuras reticulares (redes). La red (network) es una estructura más general que la jerárquica, ya que cada nodo puede tener uno o más predecesores inmediatos (así como uno o más sucesores inmediatos, igual que en la estructura jerárquica). El modelo red permite, pues, la representación directa de lazos semánticos del tipo (M:N).

Para la construcción de redes, este modelo utiliza el concepto de conjunto (set): estructura jerárquica de dos niveles. Un tipo de conjunto se compone de un tipo de registro, denominado propietario, (nodo de la estructura jerárquica de dos niveles) y uno o más tipos de registros, denominados miembros (2° nivel de la estructura jerárquica). Las reglas de definición de conjuntos permiten construir redes de gran complejidad. Así, un tipo de registro puede ser miembro de uno o más tipos de conjuntos; un tipo de registro puede ser miembro de un tipo de conjunto y propietario de otro, etc., son reglas de definición de conjuntos.

Para representar el sistema real propuesto con el modelo red, podemos definir tres tipos de registros: PILOTO, AVION y VUELO; y dos tipos de conjuntos: PI-VU (propietario: PILOTO; miembro: VUELO) y AV-VU (propietario: AVION; miembro: VUELO) como se muestra en la figura I-6.

Todos los registros miembros conectados a un registro propietario están recorridos por una cadena lógica que comienza y finaliza en dicho registro propietario. En la figura I-7 se representan los registros y cadenas lógicas para los mismos datos utilizados en el modelo jerárquico.

Básicamente, el sublenguaje de datos asociado al modelo red permite al usuario recorrer las diferentes cadenas lógicas de conexión y obte-

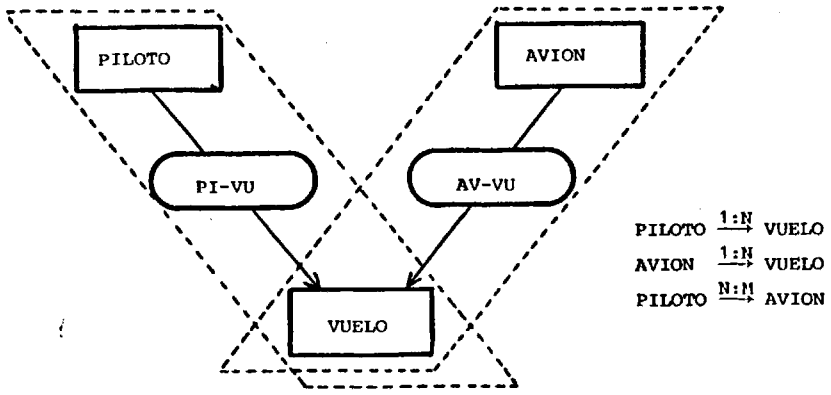


Figura I-6

ner de los nodos accedidos su información. El recorrido a través de la red se realiza por medio de las diferentes formas del comando FIND; las más importantes son:

FIND - accede de forma directa a un registro propietario.

FIND NEXT - accede secuencialmente al siguiente registro de una cadena lógica.

FIND OWNER - accede a un registro propietario desde un registro de su cadena lógica.

La obtención de la información contenida en un registro accedido por los anteriores comandos se realiza con el comando GET.

De la misma forma, un registro accedido puede modificarse o borrarse con los comandos MODIFY y DELETE. También pueden introducirse nuevos registros en nodos accedidos con el comando INSERT.

Veamos la expresión con este sublenguaje del procedimiento que responde a las dos preguntas planteadas en el modelo jerárquico.

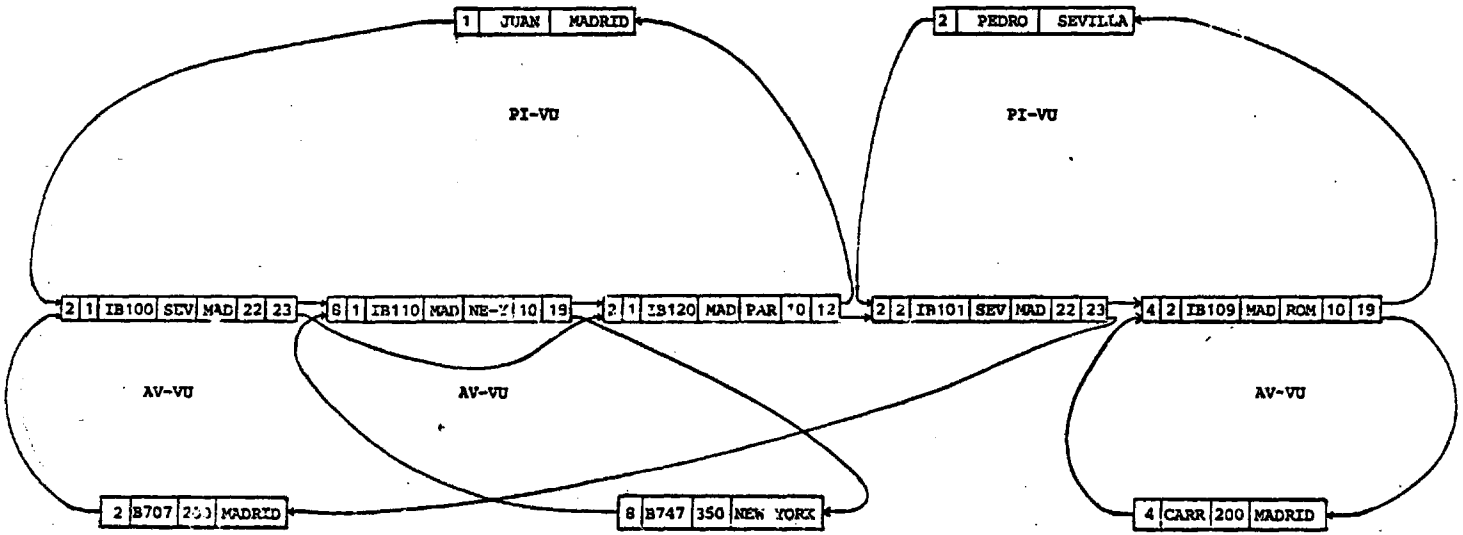
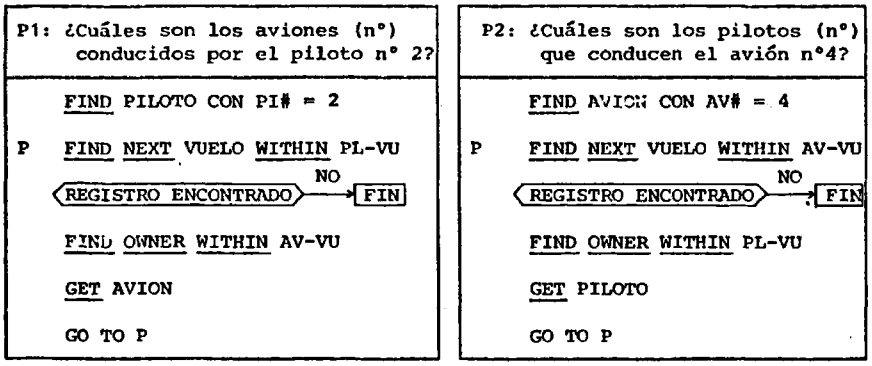


Figura I-7.- Ejemplo de modelo Red



Se puede ver que en el modelo red preguntas simétricas requieren procedimientos de respuesta simétricos, consecuencia inmediata de la posibilidad de representar directamente conexiones del tipo (N:M). Por otra parte, las dificultades de almacenamiento, puestas de manifiesto en el modelo jerárquico, desaparecen. No obstante, ello no significa la ausencia total de este tipo de problemas, pues, como veremos en el apartado I.4, se trata realmente de un problema de normalización. Es importante poner de manifiesto la complejidad que, para el usuario de la base, supone el conocimiento de todas las cadenas de conexiones de la red cuando accede a la base. El sublenguaje obliga a especificar al sistema el COMO acceder al dato demandado, en lugar de permitir, como sería más lógico, expresar QUE es lo que se desea obtener, con un lenguaje más próximo a las necesidades del usuario.

I.3.3.- Modelo relacional

El modelo relacional de datos fue introducido por Codd ([7]) en 1970 con la intención de proporcionar soluciones viables a varios problemas de la gestión de bases de datos. En particular, Codd se planteó el

problema de crear un modelo de datos que nada tuviese que ver con las diferentes condiciones de implementación y poner a disposición de los usuarios un sublenguaje de datos asociado.

Este modelo se basa en la teoría matemática de las relaciones, lo que significa que todos los resultados de esta teoría pueden aplicarse a la resolución de problemas relacionados con dicho modelo, especialmente al diseño de un sublenguaje de datos asociado.

El término relación se utiliza aquí en su sentido matemático. Dados los conjuntos D_i $(i = 1,\ldots,n)$ no necesariamente distintos, R es una relación sobre ellos si es un conjunto ordenado de n-tuples de la forma $(d_1,\ldots,d_n)$ tales que $d_i \in D_i (i=1,\ldots,n)$. Más concisamente, R es una relación sobre los M conjuntos D_i si es un subconjunto de su producto cartesiano: $R \subseteq D_i \times \ldots \times D_n$.

A los conjuntos D_i $(i=1,\ldots,n)$ se les denomina <u>dominios</u> de R. El valor de n es el grado de R. Las relaciones de grado 1 se denominan monarias, las de grado 2 binarias y, en general, las de grado n n-arias.

Una relación está <u>normalizada</u> (1NF) si sus dominios son simples, esto es, sus elementos no son a su vez conjuntos. Por razones de exposición se suele utilizar la representación tabular (matricial) de una relación normalizada, pero esta representación particular no es esencial para el modelo relacional. Por ello, hay que tener en cuenta que cada tabla representando una relación n-aria R tiene las siguientes propiedades:

1) Cada fila representa un n-tuple de R
2) El ordenamiento de las filas no es significativo
3) Todas las filas son distintas
4) El ordenamiento de las columnas es significativo si corresponde al ordenamiento de los dominios. Si cada columna se

etiqueta con el nombre del correspondiente dominio, el ordenamiento de las columnas no es significativo.

Las propiedades 2) y 3) son consecuencia inmediata del hecho de que una relación es un conjunto.

Por ejemplo, la representación tabular de una relación ternaria R(A, B, C) definida sobre los dominios A = $\{a_1, a_2, \ldots\}$, B = $\{b_1, b_2, \ldots\}$ y C = $\{c_1, c_2, \ldots\}$ podría ser:

R	A	B	C
	a_1	b_3	c_1
	a_2	b_1	c_2
	a_3	b_2	c_4
	a_8	b_7	c_{10}

La relación normalizada se utiliza en el modelo relacional para representar tanto a las entidades como a las conexiones entre ellas. Cada fila (que denominaremos simplemente tuple) de una relación normalizada representa los diferentes valores de los atributos de una entidad o conexión; la relación completa representará los valores de un conjunto de entidades o un conjunto de conexiones. Cada columna (que denominaremos dominio) representa el conjunto de valores que un atributo toma en las diferentes entidades o conexiones de la relación. Dos o más atributos de una relación podrán tomar valores de un mismo dominio; por ello, se representa en una relación el nombre de los atributos y no el de los dominios subyacentes. (En la terminología relacional usual se suele utilizar como equivalentes los términos atributo y dominio).

Una base de datos relacional es, pues, una colección de relaciones normalizadas de diferentes grados y conectadas a través de dominios comunes. Las relaciones variarán con el tiempo como resultado de la interacción de los usuarios con la base. Así, podrán eliminarse tuples, modificarse o insertarse.

El ejemplo de sistema real definido en el apartado I.3, podemos representarlo según el modelo relacional con tres relaciones:

PILOTO (PI#, PINOM, CR)

AVION (AV#, AVNOM, CAP, CB)

VUELO (VU#, PI#, CO, CD, HP, HL)

los dos primeros para los tipos de entidades piloto y avión y la tercera para la conexión entre estas entidades (vuelos) y sus atributos correspondientes.

La representación tabular de estas relaciones, para los mismos valores utilizados en los modelos jerárquico y red, será (Fig. I-8):

VUELO	VU#	PI#	AV#	CO	CD	HP	HL
	IB100	1	2	MADRID	SEVILLA	11	12
	IB101	2	2	SEVILLA	MADRID	22	23
	IB109	2	4	MADRID	ROMA	16	19
	IB110	1	8	MADRID	NEW YORK	10	18
	IB120	1	2	MADRID	PARIS	10	12

PILOTO	PI#	PINOM	CR
	1	JUAN	MADRID
	2	PEDRO	SEVILLA

AVION	AV#	AVNOM	CAP	CB
	2	B707	200	MADRID
	4	CARA	200	MADRID
	8	B747	350	NEW YORK

Figura I-8

En una BD relacional la respuesta a un pregunta dada se entiende como la formación de una nueva relación derivada de las que constituyen la base. Los sublenguajes de datos asociados a este modelo deberán, pues, permitir la definición de la relación respuesta correspondiente a una pregunda determinada. Codd propuso dos fomas de especificar tal definición:

1) Por medio de una secuencia de operaciones del algebra relacional sobre las relaciones de la base.

2) Por medio de un predicado del cálculo relacional.

La diferencia entre estos dos enfoques es análoga a la existente entre (1) definir un conjunto especificando una secuencia de operaciones (unión, intersección, etc.,) y (2) definir un conjunto por medio de una propiedad (predicado) que deben cumplir sus elementos. Será, pues, posible asociar al modelo relacional un sublenguaje de datos basado en el álgebra relacional ("procedural") o un sublenguaje basado en el cálculo relacional ("no-procedural").

I.3.3.1.- Algebra relacional

El propósito de este álgebra ([8]) es proporcionar un conjunto de operaciones sobre relaciones normalizadas de cualquier grado apropiado para la selección de datos en una base relacional. Este algebra, además de poderse utilizar como base para la definición de un sublenguaje de datos asociado al modelo relacional, constituye una herramienta teórica útil para medir la potencia selectiva de un sublenguaje de datos cualquiera.

Las operaciones relacionales podemos dividirlas en dos grupos: los tradicionales entre conjuntos (unión, diferencia, intersección y producto cartesiano) y las específicas sobre relaciones (proyección, composi-

ción, restricción y división). Las relativas a presentación de datos, tales como el ordenamiento de una relación por valores de uno de sus cominios, etc., no se tratan en este álgebra ya que este tipo de operaciones no afectan al contenido de información de los datos recuperados.

Daremos una definición precisa de cada una de las operaciones para poder, en el capítulo II, demostrar el carácter completo del repertorio de instrucciones del procesador que proponemos.

- UNION.- La unión de una relación R de grado n con otra relación S del mismo grado y dominios compatibles, es una nueva relación RUS de grado n definida por:

$$R \cup S = \{t: t \in R \vee t \in S\}$$

- INTERSECCION.- La intersección de una relación R de grado n con otra relación S del mismo grado y dominios compatibles es una nueva relación R∩S de grado n definida por:

$$R \cap S = \{t: t \in R \wedge t \in S\}$$

- DIFERENCIA.- La diferencia de una relación R de grado n con otra relación S del mismo grado y dominios compatibles es una nueva relación R - S de grado n definida por:

$$R - S = \{t: t \in R \wedge t \notin S\}$$

- PRODUCTO CARTESIANO.- El producto cartesiano de una relación R de grado n con una relación S de grado m es una nueva relación R@S de grado $n+m$ definida por:

$$R @ S = \{(\widehat{r\,s}): r \in R \wedge s \in S\}$$

siendo $\widehat{r\,s}$ la concatenación de estos dos dominios.

- PROYECCION.- Sea R una relación de grado n y L un subconjunto de m dominios de R ($m < n$). $\forall r \in R$ la notación $r[L]$ representará el conjunto de valores de los m dominios de L en el tuple r.

La proyección de la relación R sobre los m dominios de L se define como una nueva relación $R[L]$ de grado m dada por:

$$R[L] = \{r[L] : r \in R\}$$

Es decir, la relación proyección se obtiene seleccionando los dominios especificados en la definición, ordenándolos según la ordenación en L y eliminando los tuples redundantes que aparezcan.

Ejemplo

Siendo:

R	A	B	C
	a	2	f
	b	1	g
	c	3	f
	d	3	g
	e	2	g
	a	2	g

serán:

R[A]	A
	a
	b
	c
	d
	e

R[B]	B
	2
	1
	3

R[B,A]	B	A
	2	a
	1	b
	3	c
	3	d
	2	e

R[A,C]	A	C
	a	f
	b	g
	c	f
	d	g
	e	f
	a	g

- COMPOSICION (JOIN).- Siendo $\theta \in \{=, \neq, <, \leq, \geq, >\}$, la θ-composición de dos relaciones -R de grado n sobre un dominio A- y -S de grado m sobre un dominio B- es un nueva relación de grado $n+m$ definida por:

$$R[A\theta B]\ S = \{(\widehat{r\,s}) : r \in R \wedge s \in S \wedge (r[A]\ \theta\ s[B])\}$$

En la definición se supone que cada elemento de $R[A]$ es θ-comparable con cada elemento $S[B]$.

Resulta inmediato de la definición que $R[A\theta B] \subseteq R\emptyset S$ y además $R[C<D]S \cup R[C=D]S \cup R[C>D]S = R\emptyset S$.

La operación de composición más corriente es la que utiliza el operador "=", denominada equi-composición. En este caso, dos de los dominios de la relación resultante (sobre los que se realiza la composición) son idénticos; si uno de ellos se elimina por proyección el resultado es la composición natural.

Ejemplo

Siendo:

R	A	B	C
	a	1	1
	a	2	1
	b	1	2
	c	2	5
	c	3	3

S	D	E
	2	u
	3	v
	4	u

serán:

R[C=D]S	A	B	C	D	E
	b	1	2	2	u
	c	3	3	3	v

R[C>D]S	A	B	C	D	E
	c	2	5	2	u
	c	2	5	3	v
	c	2	5	4	u
	c	3	3	2	u

La θ-composición puede generalizarse a un conjunto de dominios (>1). Si $A=\{A_1,\ldots,A_p\}$ y $B=\{B_1,\ldots,B_p\}$ son dos subconjuntos de dominios de R y S respectivamente; y A_i es θ-comparable con B_i $(i=1,\ldots,p)$ la θ-composición de R y S sobre A y B se define por:

$$R[A\theta B]S = \{(r\frown s): r \in R \wedge s \in S \wedge (r[A_1]\theta\, s[B_1])\wedge\ldots\wedge(r[A_p]\theta\, s[B_p])\}$$

- RESTRICCION.- Siendo $\theta\in\{=, \neq, <, \leq, \geq, >\}$, la θ-restricción de una relación R, sobre dos de sus dominios A y B, es una nueva relación del mismo grado definida por:

$$R[A\theta B] = \{r: r \in R \wedge (r[A]\theta\, r[B])\}$$

se supone que los elementos $r[A]$ son θ-compatibles con los $r[B]$.

Ejemplo

Siendo:

R	A	B	C
	a	3	7
	b	5	4
	a	5	5
	c	1	2

serán:

R[B=C]	A	B	C
	a	5	5

R[C>B]	A	B	C
	a	3	7
	c	1	2

De la misma forma que la θ-composición, la θ-restricción se puede generalizar a dos conjuntos de dominios. Siendo $A=\{A_1,\ldots,A_p\}$ y $\{B=B_1,\ldots,B_p\}$ con A_i θ-comparable con B_i y A_i,B_i dominios de R $(i=1,\ldots,p)$ la θ-restricción de R sobre A y B se define por:

$$R[A\theta B] = \{r: r \in R \wedge (r[A_1]\theta\, r[B_1])\wedge\ldots\wedge(r[A_p]\theta\, r[B_p])$$

- DIVISION.- La división relacional es una operación entre dos relaciones (dividendo y divisor) que da como resultado una tercera relación (cociente). Por motivos de claridad definiremos, en primer lugar, la división en su forma más simple, esto es, cuando el dividendo es una relación binaria y el divisor y cociente monarias.

Siendo $R(A,B)$ la relación dividendo, $S(C)$ la relación divisor y los dominios B y C compatibles, se define la relación cociente $T(A)$ (sobre el primer dominio de R) como:

$$T(A) = R(A,B) \div S(C) = \{a: (a,b) \in R \;\forall\; b \in S\}$$

es decir, un valor a perteneciente al dominio A de R pertenece a la relación cociente, sí y sólo si para todos los valores b_i pertenecientes al dominio del divisor (c) existen pares (a,b_i) pertenecientes a la relación dividendo. Definiendo el conjunto imagen de un valor $a \in A$ sobre R como:

$$g_R(a) = \{b: (a,b) \in R\}$$

la relación cociente $T(A)$ puede expresarse como:

$$T(A) = R(A,B) \div S(c) = \{a: a \in A \wedge S(c) \subseteq g_R(a)\}$$

Ejemplo:

Siendo:

R	A	B
	1	a
	1	b
	1	c
	1	d
	1	e
	2	a
	3	e
	4	a
	4	e
	5	e
	3	f

S	C
	e

U	C
	a
	e

Z	C
	a
	b
	e

Y	C
	a
	b
	f

serán:

R÷S	A
	1
	3
	4
	5

R÷U	A
	1
	4

R÷Z	A
	1

R÷Y	A
	θ

relación vacía

Para generalizar la división al caso de una relación dividiendo R de grado n, generalizaremos previamente la definición de conjunto imagen sobre R, y para ello introduciremos la siguiente terminología. Si R es una relación de grado n definida sobre un conjunto D de dominios y L es un subconjunto de D, $\overline{L}$ designará el complemento de L respecto de D, es decir,

$$(L \cup \overline{L} = D) \wedge (L \cap \overline{L} = \emptyset)$$

El conjunto imagen de una serie de valores - pertenecientes a dominios de R- sobre R, se define por:

$$g_R\,(r[L]) = \{r[\overline{L}] : (t[L], r[\overline{L}]) \in R\}$$

es obvio de la definición que $g_R(r[\overline{L}]) \subseteq R[L] \quad \forall r \in R$

Siendo R una relación de grado m, S una relación de grado n, A un subconjunto de los dominios de R, B un subconjunto de los dominios de S y las proyecciones $R[A]$ y $S[B]$ compatibles, la división de -la relación R sobre A- entre -S sobre B- se define por:

$$R[A \div B]\ S = \{r[\overline{A}] : r \in R \wedge S[B] \subseteq g_R\,(r[\overline{A}])\}$$

Ejemplo:

Siendo:

R	A	B	C
	1	9	a
	2	9	b
	3	9	c
	4	0	a

S	D	F
	a	1
	a	2
	b	1

será:

R[BC] [C÷D] S	F
	9

Para poner de manifiesto la utilización del álgebra relacional, pasamos a expresar, en términos de sus operaciones, las dos preguntas formuladas en anteriores modelos:

P1: ¿Cuáles son los aviones (n°) conducidos por el piloto n° 2?

Definiendo la relación constante PI2 como

PI2	PI#
	2

la expresión relacional sería:

VUELO [PI# = PI#] PI2 [AV#]

es decir, realizamos la composición de las relaciones VUELO y PI2 sobre el dominio PI# en ambas y con el operador =; después proyectamos la relación resultante sobre el dominio AV#. (Hemos supuesto tácitamente que la composición es de mayor prioridad que la proyección en la evaluación de una expresión relacional).

La relación respuesta sería:

AV#
2
4

(para los datos de la figura I-8).

P2: ¿Cuáles son los pilotos (n°) que conducen el avión n° 4?.

Definiendo la relación constante AV4 como

AV4	AV#
	4

la expresión relacional sería:

VUELO [AV# = AV#] AV4 [PI#]

y la respuesta:

PI#
2

Puede observarse, además de la simetría de las expresiones, la concisión que el álgebra relacional presenta en la formulación de accesos. La inserción y eliminación de tuples a una relación puede realizarse con las operaciones de unión y diferencia, respectivamente de esta relación con relaciones constantes que contengan los tuples insertados o eliminados. La modificación puede efectuarse por una secuencia de las operaciones diferencia y unión.

I.3.3.2.- Cálculo relacional

El cálculo relacional es una notación matemática, basada en el cálculo de predicados, que sirve para definir una relación en forma intensiva a partir de las existentes en la base.

La forma general de una expresión de este cálculo es: $\{L:P\}$ donde P es un predicado que define un conjunto de tuples (una relación) y L una lista objeto de dominios -con indicación de la relación de procedencia- sobre los cuales se proyecta la relación global definida por P. En general, un predicado puede ser de una complejidad arbitraria, pero, formulado de acuerdo con las reglas usuales ([8]). Los operadores utilizados en la formulación de un predicado son los de comparación ($=$, $\neq$, $<$, $\leq$, $\geq$, $>$) y los booleanos ($\wedge$, $\vee$, $\neg$). Se utilizan paréntesis para alterar el orden de evaluación normal, de izquierda a derecha, de los operadores. La utilización, en la formulación de predicados, de tuples variables para conectar relaciones y cunatificadores (existencial $\exists$ y universal $\forall$) para cuantificar su rango de variación, confiere a las expresiones del cálculo relacional una alta potencia selectiva.

Para poner de manifiesto la formulación de expresiones, veamos la forma que adoptan, en cálculo relacional, las respuestas a las dos pre-

guntas planteadas en anteriores apartados.

P1: ¿cuáles son los aviones (n°) conducidos por el piloto n° 2?

{VUELO.AV# : VUELO.PI# = '2'}

El predicado de esta expresión selecciona de la relación VUELO todos los tuples cuyo atributo PL# (n° de piloto) es 2. La lista objeto proyecta estos tuples sobre el atributo AV# (n° de avión).

P2: ¿Cuáles son los pilotos (n°) que conducen el avión n° 4?

{VUELO.PI# : VUELO.AV# = "4"}

cuya interpretación es análoga a la anterior.

Preguntas más complejas requieren el uso de cuantificadores, así, la pregunta: ¿Cuáles son los nombres de los pilotos que cubren los vuelos entre Madrid y Sevilla?, se formularía:

{PILOTO.PINOM : ∃ VUELO (VUELO.PI# = PILOTO.PI# ∧ VUELO.CO =
= 'MADRID' ∧ VUELO.CD = 'SEVILLA'}

En este caso, para que un tuple de relación PILOTO sea seleccionado por el predicado se le exige que exista un tuple en la relación VUELO que, además de tener igual valor que aquél en el atributo PI#, contenga los valores Madrid y Sevilla en los atributos respectivos CO y CD.

Un sublenguaje de datos capaz de expresar cualquier pregunta cuya información de respuesta esté semánticamente contenido en la base se dice que es completo. Codd ha propuesto una medida de la "completitud" (potencia selectiva) de un sublenguaje de datos: "completitud relacional", y llega a la conclusión de que el cálculo es relacionalmente completo. Además, demuestra que cualquier pregunta expresable en cálculo es también expresable en álgebra, es decir, el álgebra relacional tiene, al menos, la misma poten-

cia selectiva que el cálculo ([8]). Esta propiedad la utilizaremos en el Capítulo II para demostrar la "completitud" relacional del repertorio de instrucciones del procesador que proponemos.

Por lo que respecta a la utilización, el cálculo, al ser un sublenguaje no-"procedural", presenta frente al álgebra un mayor grado de independencia de datos, así como una mayor legibilidad de las expresiones respectivas.

Un sublenguaje de datos basado directamente en el cálculo relacional fue propuesto por Codd con el nombre de lenguaje ALPHA ([9]). Presenta, además de todas las características selectivas del cálculo, la posibilidad de utilizar funciones de agregación tales como "promedio", "máximo", "mínimo", etc., en la formulación de predicados. Otros sublenguajes de datos propuestos tales como SEQUEL ([10]), SQUARE ([11]), etc., están basados, de forma más o menos directa, en el cálculo relacional.

I.4.- NORMALIZACION

Para que una BD relacional tenga la máxima protección frente a las operaciones de almacenamiento y cambios futuros de su contenido, no es suficiente que sus relaciones sean normales en el sentido que precisamos en su definición (1NF). La presencia de restricciones semánticas en sus datos hace necesario definir nuevas formas normales. El proceso de normalización que estudiaremos para bases relacionales es aplicable, también, a bases no relacionales que agrupen sus ítems de datos en estructuras planas como, por ejemplo, los segmentos del DBTG.

Cuando las relaciones de una base cumplen la definición dada en

el apartado I.3.3., se dice que están en su primera forma normal (1NF). Esta forma elimina todos los dominios no simples. En lo que sigue, definiremos la segunda (2NF) y tercera (3NF) formas normales introducidas por Codd ([13]), ([14]).

I.4.1.- Dependencia funcional

En una relación R diremos que un atributo B es funcionalmente dependiente de otro atributo A, si en todo instante de tiempo cada valor en A tiene asociado un valor único de B en todos los tuples de R. Esta dependencia funcional entre dos atributos la representaremos por $R.A. \longrightarrow R.B.$ Si B no depende funcionalmente de A escribiremos $R.A \not\longrightarrow R.B.$ Y si se cumple $(R.A \longrightarrow R.B) \wedge (R.B \longrightarrow R.A)$ en todo instante de tiempo A y B mantienen una correspondencia biunívoca y escribiremos $R.A \longleftrightarrow R.B.$

La definición anterior puede generalizarse a subconjuntos de atributos (atributos compuestos). Así, si D y F son dos subconjuntos diferentes de atributos de R, diremos que E es funcionalmente dependiente de D si en todo instante de tiempo cada valor $r[D]$ tiene un vamor único $r[E]$ asociado $\forall\, r \in R$. La dependencia funcional $R.D \longrightarrow R.E.$ cuando $E \subset D$ la llamaremos dependencia trivial.

El reconocimiento de las dependencias funcionales en una base relacional es esencial para entender el significado o semántica de los datos. Así, en la relación:

R (E#, DE#, DI#) donde E# = nº de empleado
DE# = nº de departamento
DI# = nº de división

la dependencia funcional $R.E\# \longrightarrow R.DE\#$ significa que cada empleado perte-

nece a un único departamento.

Las dependencias funcionales representan restricciones semánticas de los datos del sistema real que se modelan en una BD.

Si en el ejemplo anterior, además, cada departamento nunca pertenece a más de una división, se darán las siguientes dependencias funcionales:

R.E# ⟶ R.DE# (vista anteriormente)

R.DE# ⟶ R.DI#

R.E# ⟶ R.DI# (consecuencia de las dos anteriores)

R.(E#,DE#) ⟶ R.DI# (consecuencia de la anterior)

Si además suponemos que normalmente existen muchos empleados pertenecientes a un departamento dado y muchos departamentos pertenecientes a una división dada, se cumplirá:

R.DE# ⟶/ R.E#

R.DI# ⟶/ R.DE#

Como ejemplo de dependencia trivial tendríamos: R.(E#,DE#)⟶ R.E#

Todas las dependencias funcionales del ejemplo anterior podemos representarlas gráficamente, como muestra la figura I.9.

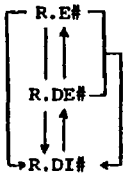


Figura I-9

I.4.2. Claves candidatas y clave primaria

Una clave candidata K de una relación R es, por definición, todo

subconjunto de atributos (en particular puede ser uno solo) de R que cumple las siguientes propiedades:

1.- Identificación única, esto es, en cada tuple de R el valor de K lo identifica unívocamente. En términos de dependencia funcional $R.K \longrightarrow R.\Omega$ donde Ω es el conjunto de todos los atributos de R.

2.- No redundancia, esto es, ningún atributo de K puede suprimirse sin dejar de cumplirse la propiedad 1.

Es obvio que en una relación R siempre existirá, al menos, una clave candidata ya que, por definición de relación, el conjunto de todos los atributos de R poseen la propiedad 1.

De entre todas las claves candidatas de una relación R se elige arbitrariamente una, llamada clave primaria, para identificar sus tuples.

En la relación R(E#, DE#, DI#), vista anteriormente, la única clave candidata será el atributo E# y, por tanto, será la clave primaria de R. La clave primaria de una relación se suele indicar subrayando los atributos que la componen.

A todo atributo que participe de una clave candidata le llamaremos atributo primo. Los restantes atributos serán atributos no-primos. Así, en la anterior relación, el único atributo primo es E#; siendo DE# y DI# atributos no-primos.

I.4.3.- Dependencia funcional completa y parcial

Supongamos que D y E son dos subconjuntos diferentes de atributos de la relación R que cumplen: $R.D. \longrightarrow R.E.$ Diremos que E es completamente dependiente de D en R, si E no es funcionalmente dependiente de ningún subconjunto de D (eliminando obviamente el propio D).

Formalmente, podemos decir que E es completamente dependiente de D sí y sólo si $\forall\ S \subset D \wedge S \neq D$ se cumple $(R.D \longrightarrow R.E) \wedge (R.S \not\longrightarrow R.E)$.

Si se cumple que $(R.D \longrightarrow R.E) \wedge (R.T \longrightarrow R.E) \wedge (TCD) \wedge (T \neq D)$ diremos que el dominio compuesto E es parcialmente dependiente de D.

Con objeto de ilustrar el concepto de dependencia funcional completa y motivar la definición de segunda forma normal (2NF) consideremos la siguiente relación:

R(S#, PI#, PR, CA, CS)

donde:

S# = nº de suministrador

PI# = nº de pieza que suministra

PR# = nº de proyecto al que suministra

CA = cantidad suministrada

CS = ciudad donde suministra

Es decir, el tuple (t, u, v, x, z) pertenece a R, si el suministrador nº t suministra la pieza nº u al proyecto nº v en la cantidad x, siendo su ciudad de suministro la ciudad z.

Además supondremos que se dan las siguientes dependencias y no-dependencias funcionales:

(1) R.(S#, PI#, PR#) ⟶ R.CA (Para una combinación de estos tres atributos existe una única cantidad)

(2) R.S# ⟶ R.CS (Un suministrador suministra en una única ciudad)

(3) R.(S#, PI#) ⟶ R.PR# (Una combinación de S#, PI# puede ser asociada con más de un proyecto)

(4) R.(PI#, PR#) ⟶ R.S# (Una combinación de PI#, PR# puede ser asociada con más de un suministrador).

(5) R.(PR#, S#) $\not\longrightarrow$ R.PI# (Una combinación de PR#, S# puede ser asociada con más de una pieza)

(6) R.(S#, PI#) $\not\longrightarrow$ R.CA (La cantidad no viene determinada por S#,PI#)

(7) R.(PI#, PR#) $\not\longrightarrow$ R.CA (La cantidad no viene determinada por PI#,PR#)

(8) R.(PR#, S#) $\not\longrightarrow$ R.CA (La cantidad no viene determinada por PR#,S#)

De (2) se deduce que R.(S#, PI#, PR#) $\longrightarrow$ R.CS. Esta dependencia funcional junto con (1), (6), (7) y (8) determinan el carácter de clave candidata para la combinación de atributos (S#, PI#, PR#). Al ser, además, única, se trata también de la clave primaria: R(S#,PI#,PR#,CA,CS). Los atributos primos será, pues, S#, PI# y PR#; y los no-primos CA y CS. La dependencia funcional (1) es completa; la dependencia R.(S#, PI#, PR#) $\longrightarrow$ R.CS es parcial, pues, al cumplirse (2), CS depende funcionalmente de un subconjunto de los atributos (S#, PI#, PR#).

Este tipo de dependencias parciales de atributos no-primos, como CS, de una clave primaria, como (S#, PI#, PR#) y, en general, de cualquier clave candidata, introducen las siguientes anomalías de almacenamiento (supondremos que la relación R representa en un cierto instante de tiempo los valores mostrados en la figura I-10):

R	S#	PI#	PR#	CA	CS
	S1	PI1	PR1	3	C1
	S1	PI2	PR2	2	C1
	S1	PI3	PR2	2	C1
	S2	PI1	PR1	3	C2
	S2	PI2	PR2	2	C2

Figura I-10

- Inserción: Si se desea introducir la información correspondiente a la ciudad C3, donde operará el suministrador S3, ello no será posible hasta que se asigne alguna pieza a suministrar, a un determinado proyecto y en una determinada cantidad.
- Borrado: Si el suministrador S2 cesa de suministrar las piezas PI1 y PI2 y se eliminan los tuples correspondientes, se perderá la información relativa a su ciudad de operaciones.
- Actualización: Si se asigna a un suministrador una nueva ciudad para realizar sus operaciones, en general, más de un tuple debe ser actualizado con el consiguiente riesgo de inconsistencia.

Estas anomalías de almacenamiento se resuelven introduciendo la segunda forma normal.

I.4.4.- Definición de segunda forma normal (2NF)

Una relación R diremos que está en la segunda forma normal (2NF) si lo está en la primera (1NF) y cada uno de sus atributos no-primos mantienen una dependencia funcional completa sobre cada clave candidata de R.

La conversión de una relación 1NF a otras 2NF se realiza sustituyendo la primera por proyecciones de ésta sobre determinados atributos, de forma que se eliminen las dependencias parciales de atributos no-primos. Así, la relación R(S#, PI#, PR#, CA, CS) anteriormente estudiada, podemos sustituirla por las proyecciones $R1 = R[R\#, PI\#, PR\#, CA]$ y $R2 = R[R\#, CS]$ (figura I-11), que como puede observarse, no presentan las anteriores anomalías por encontrarse en la 2NF. El contenido semántico de R1 y R2 sigue siendo el mismo que tenía R.

R1	S#	PI#	PR#	CA
	S1	PI1	PR1	3
	S1	PI2	PR2	2
	S1	PI3	PR2	2
	S2	PI1	PR1	3
	S2	PI2	PR2	2

R2	S#	CS
	S1	C1
	S2	C2

Figura I-11

I.4.5.- Dependencia transitiva

Si A, B y C son tres conjuntos de atributos de una relación R de grado mayor o igual que tres y se cumple:

$$R.A \longrightarrow R.B$$

$$R.B \not\longrightarrow R.A$$

$$R.B \longrightarrow R.C$$

diremos que C es transitivamente dependiente de A en la relación R.

Las dependencias transitivas introducen en una BD anomalías de almacenamiento que vamos a poner de manifiesto estudiando la siguiente relación:

R(E#, D#, J#, TC) donde: E# = nº de empleado (diferente para cada uno)

D# = nº de departamento donde trabaja el empleado

J# = nº del jefe del empleado

TC = tipo de contrato (oficial o no oficial) bajo el que se desarrolla el trabajo que el departamento, al que pertenece el empleado, realiza.

con las siguientes dependencias y no dependencias funcionales:

(1) R.E# ⟶ R.D# (cada empleado pertenece a un único departamento)

(2) R.D# ⟶/ R.E# (cada departamento tiene más de un empleado)

(3) R.D# ⟶ R.J# (cada departamento tiene un único jefe)

(4) R.J# ⟶ R.D# (cada jefe lo es de un único departamento)

(5) R.D# ⟶ R.TC (cada departamento trabaja bajo un único tipo de contrato)

Todas estas dependencias, junto con las que de ellas se deducen (exceptuando las triviales) podemos representarlas como muestra la figura I-12.

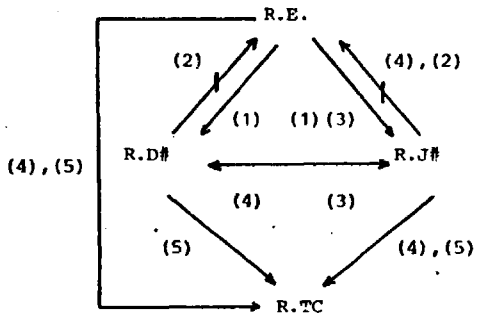


Figura I-12

De este esquema de dependencias se deducen las siguientes dependencias transitivas:

TC es transitivamente dependiente de E#; consecuencia de (1), (2) y (5)

D# es transitivamente dependiente de E#; consecuencia de (1)(3), (4)(2) y (4)(5)

J# es transitivamente dependiente de E#; consecuencia de (1), (2) y (3)

Si suponemos que R presenta en un cierto instante de tiempo los

valores mostrados en la figura I-13, analicemos las anomalías de almacenamiento:

R	E#	D#	J#	TC
	1	d1	11	OFIC
	2	d1	11	OFIC
	3	d2	12	NOOF
	4	d1	11	OFIC
	5	d2	12	NOOF
	6	d2	12	NOOF
	7	d3	13	NOOF

Figura I-13

- Inserción: Si se desea introducir información relativa al departamento n° d4, como puede ser su número, jefe o tipo de contrato bajo el que trabajará, ello no será posible hasta que exista algún empleado adscrito a dicho departamento.

- Borrado: Si se produce la baja del empleado n° 7, eliminándose el correspondiente tuple de la relación, se pierde toda la información referente al departamento al que pertenecía (D#, J# y TC).

- Actualización: Si el jefe de un departamento cambia, todos los tuples correspondientes a los empleados que trabajan en ese departamento (en general. más de uno) deben actualizar el dominio J#. Lo mismo ocurre si se altera el tipo de contrato (TC) o n° de departamento (D#). La probabilidad de inconsistencia aumenta.

La introducción de una nueva forma normal, la tercera (3NF), superará estas anomalías.

I-4.6.- Definición de tercera forma normal (3NF)

Una relación R está en la tercera forma normal (3NF) si lo está en la 2NF (y por tanto en la 1NF) y cada uno de sus atributos no-primos, no dependen transitivamente de cada clave candidata de R.

Obviamente, la relación R(E#, D#, J#, TC), estudiada en el anterior apartado, no está en la 3NF ya que, los atributos D#, J# y TC dependen transitivamente de la única clave candidata (clave primaria) E#. Es obvio, también, que R está en la 2NF, puesto que al estar la única clave candidata compuesta por un único atributo, no pueden existir dependencias parciales.

El paso de la 2NF a la 3NF se realiza por proyección sobre determinados atributos, de forma que desaparezcan todas las dependencias transitivas. Así, la relación R es semánticamente equivalente a $R1 = R[E\#, D\#]$ y $R2 = R[D\#, J\#, TC]$ que están en la 3NF. La figura I-14 muestra los valores correspondientes.

R1	E#	D#
	1	d1
	2	d1
	3	d2
	4	d1
	5	d2
	6	d2
	7	d3

R2	D#	J#	TC
	d1	11	OFIC
	d2	12	NOOF
	d3	13	NOOF

Figura I-14

Un procedimiento para sintetizar algorítmicamente una BD relacional, con todas sus relaciones en la 3NF, conteniendo el menor nº de ellas y partiendo de las dependencias funcionales, se describe en [14].

I.5.- INTEGRIDAD DE UNA BASE DE DATOS

La integridad de una BD refiere la exactitud semántica de sus datos, esto es, el cumplimiento de un conjunto de restricciones impuestas por la naturaleza del sistema real que modela. Así, una BD que almacena información relativa a las actividades de una empresa que sólo emplea titulados superiores, viola su integridad cuando, por ejemplo, aparece en el atributo "titulación" de la entidad "empleado" el valor "ingeniero técnico".

Las pérdidas de integridad en una BD pueden estar cuasadas por fallo del hardware, del software o por errores en las interacciones (accesos). El sistema de control de integridad, con frecuencia, sólo se ocupa de las violaciones de integridad ocasionadas por sus usuarios. Este control de lleva a efecto permitiendo, en la fase de creación de la base, la especificación de las restricciones semánticas de sus datos por medio de un conjunto de declaraciones (declaraciones de integridad). El sistema se encargará de rechazar cualquier acceso de actualización (modificación, inserción y eliminación) que viola estas declaraciones.

Las declaraciones de integridad pueden referirse a estados válidos de la base (declaraciones de estado) o a transiciones válidas (declaraciones de transición)([15]). Por ejemplo, la declaración "ningún empleado puede ganar menos de 10.000", es una declaración de estado; en cambio, la declaración "ningún empleado puede sufrir una disminución de su salario", es una declaración de transición.

Por otra parte, las declaraciones de integridad pueden referirse a datos reales de la base o a funciones de agregación sobre datos reales. Los dos ejemplos anteriores pertenecen al primer tipo. En cambio, la declaración de estado "el salario promedio de los empleados de cualquier departamento no puede exceder de 20.000", y la declaración de transición "el sa-

lario máximo de un empleado no puede aumentar más de un 10% anual" son ejemplos de declaraciones de integridad referidas a funciones de agregación.

En general, las declaraciones de integridad sólo pueden verificarse después que el correspondiente acceso de actualización ha afectado al contenido de la base (verificación post-ejecución). Si el acceso viola alguna declaración de integridad, el sistema tendrá que recuperar el estado de la base previo a la realización del mismo. Las declaraciones de integridad de estado sobre datos reales de la base pueden verificarse antes que el acceso modifique los datos (verificación pre-ejecución) evitándose los procesos de recuperación.

1.6.- SEGURIDAD DE UNA BASE DE DATOS

En general, el concepto de seguridad de una BD se refiere a la protección de sus datos contra cualquier tipo de alteración o consulta desautorizada. Las causas que pueden atentar contra la seguridad de una BD pueden ser muy variadas. Así, fallos de hardware o software pueden provocar pérdidas irreparables de información. No obstante, el sistema de control de seguridad en una BD se ocupa, en particular, de garantizar el control selectivo de accesos, relegando la responsabilidad de los demás aspectos de la seguridad a otras funciones del sistema.

El rango de flexibilidad requerido por un sistema de control de accesos es, en general, bastante amplia. Así, habrá usuarios que puedan acceder a la totalidad de los datos de la base para cualquier tipo de operación; otros tan sólo tendrán acceso a una parte, también para cualquier tipo de operación; otros podrán consultar una parte sin posibilidad de cambiarlos o insertar datos nuevos. En ocasiones, la naturaleza de la base pue-

de exigir controles a nivel de valores concretos de los datos consultados. Así, a determinados usuarios les estará permitida la consulta de determinados datos si sus valores están comprendidos en un determinado intervalo.

Una BD con control selectivo de accesos ha de disponer, pues, de un sistema de identificación de usuarios. La sofisticación de estos sistemas dependerá del grado de seguridad requerido en cada caso particular. No obstante, en la mayoría de los casos se compone de un proceso de identificación inicial o condicional en el que el usuario declara al sistema determinada clave, seguido de un proceso de autentificación en el que el usuario debe responder a determinadas preguntas planteadas por el sistema, por ejemplo, el valor que resulta de aplicar determinada función matemática a un número aleatorio generado por el sistema.

Cuando un usuario supera con éxito ambos procesos, queda en disposición de utilizar la base con las restricciones de acceso a él asociadas. Las restricciones de acceso son definidas en la fase de creación de la base o cuando un nuevo usuario entra a formar parte de la comunidad que de ella se sirve.

REFERENCIAS

(1) DATE, C.J., "An Introduction to Data base Systems". Addison Wesley, 1976.

(2) MARTIN, J., "Organización de las Bases de Datos", Prentice-Hall International, 1977.

(3) Information Management System/360, Version 2. General Information Manual. IBM Form n° GH20-0765.

(4) Information Management System/360, Version 2. Application Programming Reference Manual. IBM Form n° SH20-0912.

(5) CODASYL Systems Committee, "Feature Analysis of Generalized Data Base Management Systems". Technical Report. (May 1971).

(6) CODASYL Systems Committee, "Introduction to Feature Analysis of Generalized Data Base Management Systems". (April 1971).

(7) CODD, E.F., "A Relational Model of Data for Large Shared Data Banks". CACM Vol. 13, n° 6, June 1970. pp. 377-387.

(8) CODD, E.F., "Relational Completeness of Data Base Sublanguages" In Data Base Systems, Courant Computer Science Symposia Seres, Vol. 6, Prentice-Hall, 1972.

(9) CODD, E.F., "A Data Base Sublanguage Founded on the Relational Calculus". Proc. 1971 ACM SIGFIDENT Workshop on Data Description, Access and Control.

(10) CHAMBERLIN, D.D. and BOYCE, R.F., "SEQUEL: A structured English query language". Proc. 1974 ACM SIGFIDENT Workshop, Ann. Arbor, Michigan.

(11) BOYCE, R.F., CHAMBERLIN, D.D. and KING III, W.F., "Specifying Queries as relational expressions": the SQUARE Data sublanguages". CACM, vol. 18 n° 11, Nov. 1975, pp. 621-628.

(12) CODD, E.F., "Normalized Data Base Structure: A Brief Tutorial". Proc. 1971 ACM SIGFIDENT Workshop on Data Description, Access and Control.

(13) CODD, E.F., "Further Normalization of the Data Base Relational Model". In Data Base Systems, Courant Computer Science Symposia Series, Vol. 6, Prentice-Hall 1972.

(14) BERNSTEIN, P.A., "Synthesizing Third Normal Form Relations from Functional Dependencies". ACM TODS Vol. 1, n° 4, December 1976, pp. 277-298.

(15) WON KIM, "Relational Database Systems". ACM Computing Surveys, Vol. II, n° 3, September 1979, pp. 185-211.

C A P I T U L O II

CONCEPCION, ARQUITECTURA Y ORGANIZACION GENERAL DEL PROCESADOR

II.1.- MAQUINAS PARA BASES DE DATOS

La utilización de ordenadores y dispositivos de memoria secundaria convencionales para soportar SGBDs plantea serias dificultades a la hora de incorporar los nuevos conceptos que, en torno a tales sistemas, se han venido desarrollando en los últimos años: modelos y sublenguajes de datos de alto nivel, integridad, seguridad, etc.

Estas dificultades provienen de una incompatibilidad esencial entre la arquitectura del ordenador convencional y las exigencias de la gestión de BD. No hay que olvidar que la concepción de la arquitectura tipo Von Neumann de los ordenadores convencionales obedeció a exigencias en la automatización del cálculo numérico y consecución de autonomía en la toma de decisiones lógicas que este cálculo conlleva. A pesar de esta filosofía de origen, los ordenadores se han utilizado en la resolución de problemas esencialmente no numéricos como son los planteados por una BD. Para ello, ha sido necesario expresar, en términos de operaciones elementales de cálculo y un sistema de referenciación de datos posicional, procesos no numéricos, fundamentalmente de búsqueda, que exigen sistemas de referenciación de datos asociativos o basados en el contenido. Esta forma de proceder, impuesta por la filosofía de funcionamiento del ordenador convencional, origina la aparición de problemas colaterales, extrínsecos a la propia naturaleza de una BD, que exigen una alta complejidad en el software que los resuelve y que limitan las posibilidades de prestación de los sis-

temas, especialmente aquéllas que dependen de una alta velocidad de respuesta (tiempo real).

La reciente aparición de las llamadas máquinas o procesadores de BD pretende dar una solución, desde el campo de la arquitectura de ordenadores, a las limitaciones de los SGBDs impuestas por la naturaleza del hardware convencional. Se trata de procesadores especializados en la gestión de BDs cuya concepción es hoy posible debido a dos razones fundamentales: a) el desarrollo experimentado en los últimos años por la tecnología hardware, especialmente en el campo de la integración de circuitos; y b) la clarificación y formalización, llevada a cabo en la última década, de las exigencias propias de una BD, traducidas en la formulación de modelos lógicos de datos y sublenguajes asociados independientes de los detalles de implementación y que expresan, de forma concisa y directa, la problemática específica del almacenamiento y recuperación de información.

Varios han sido los procesadores de BD propuestos en los últimos años. Algunos sólo son proyectos teóricos que no han sido aún implementados; otros se han estudiado a nivel de simulación para cuantificar los parámetros de "performances", y de muy pocos se han construido prototipos.

Teniendo en cuenta que la transmisión de datos, entre memoria secundaria y ordenador, es uno de los factores que más inciden en el bajo rendimiento de los sistemas convencionales, los trabajos tendentes a dotar la CPU de un ordenador con recursos especiales para ejecutar más eficientemente las funciones de BD, no han conseguido resultados muy satisfactorios. Fundamentalmente, estas investigaciones se han desarrollado en dos frentes bien diferenciados: a) el de la microprogramación, definiendo repertorios de instrucciones ajustados a las características específicas de la gestión de datos y b) el de las memorias asociativas, formando parte de la estruc-

tura operacional del ordenador a fin de facilitar los procesos de búsqueda.

La tendencia general, en la casi totalidad de los procesadores de BD que presentan mejoras substanciales de "performances", ha sido justo la contraria: delegar funciones relacionadas con la gestión de datos a procesadores, más o menos especializados, asociados a la memoria secundaria que soporta la información de la base. Desde esta perspectiva, el desarrollo de las máquinas de BD podemos verlo como un intento de elevar el nivel semántico de la interfase de comunicación de la memoria secundaria respecto de la CPU del ordenador principal que la gestiona. De esta forma, los protocolos de comunicación expresarán, de forma más directa, las necesidades finales de los accesos a la base.

La medida del desnivel semántico entre ordenador y memoria secundaria, o lo que sería equivalente, el número e importancia de las funciones delegadas por la CPU, permitiría situar a las diferentes máquinas propuestas dentro de un espectro que iría desde las memorias secundarias que incorporan en su controlador simples funciones de detección y corrección de errores, posicionamiento de cabezas, etc.([1]), hasta los procesadores de BD que incorporan las funciones más complejas de seguridad e integridad ([2]), pasando por las llamadas memorias secundarias inteligentes, capaces de realizar funciones parciales de BD ([3])([4])([5]).

Respecto a los valores de performances (tiempo de respuesta y capacidad de procesamiento) que una máquina de BD es capaz de alcanzar, un factor más importante que el comportamiento funcional externo es el grado de especialización de su organización interna. En efecto, cuanto más directo sea el planteamiento de los objetivos finales que se tratan de conseguir, menores serán los procedimientos intermedios de correspondencia

software que haya que utilizar. En este aspecto y debido a la escasa información que se suministra, resulta difícil una clasificación seria de las alternativas propuestas. Tan sólo se pueden entresacar principios generales que, en cierta medida, han inspirado la organización de un buen número de procesadores. De algunos de estos principios -de los que participa el procesador que proponemos- se tratará en el apartado siguiente.

II.2.- IDEAS GENERALES QUE INSPIRAN LA ORGANIZACION DEL PCBD

Los principios generales en torno a los cuales se ha desarrollado la organización del Procesador Concurrente de Base de Datos (PCBD) que proponemos, pretenden conseguir, dentro de las posibilidades actuales de la tecnología hardware, dos objetivos fundamentales:

- Implementar las funciones de gestión de BD siguiendo la alternativa, conceptualmente precisa, presentada por el modelo relacional.
- Conseguir mayor adecuación a la naturaleza multiusuario de una BD que la alcanzada por otros procesadores propuestos.

II.2.1.- Concepto de "back-end"

La idea de descentralizar las funciones de gestión de BD del ordenador principal para soportarlas sobre un procesador independiente, denominado "back-end", fue propuesta por Canaday en el sistema experimental XDMS ([6]). Un "back-end" es un procesador semiautónomo acoplado a un ordenador de propósito general (host) y que tiene acceso a una memoria secundaria sobre la que reside una BD (figura II-1). Su función es la de proporcionar servicios de gestión de BD al ordenador principal. El término "back-

end" se utilizó por analogía con los ordenadores "front-end" que sirven de interfase entre un ordenador principal y sus entradas externas (terminales, redes, etc.).

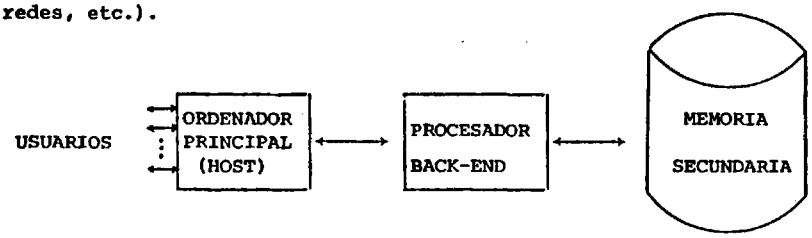


Figura II-1. Configuración back-end

La utilización de un procesador back-end libera a la CPU del ordenador principal de las conmutaciones de trabajos (switching task) necesarias para iniciar las múltiples operaciones de E/S implícitas en las funciones de gestión de BD. El ordenador principal dispondrá de más tiempo para realizar sus funciones tradicionales de preparación y ejecución de programas. Por otra parte, el "back-end" podrá especializarse al objeto de conseguir más alto rendimiento en la realización de sus funciones. Ya en el sistema XDMS, Canaday utilizó un miniordenador microprogramable (META-4) que le permitía ensayar diferentes tipos de instrucciones y ajustar así su repertorio a las tareas propias de la gestión de datos. Mientras el XDMS demostró la conveniencia de la alternativa "back-end", sus mejoras futuras de "performances" estaban muy limitadas como consecuencia de su arquitectura SISD (Single Instruction, Single Data).

El PCBD, siguiendo esta idea de la descentralización, se organiza como un procesador "back-end" de un ordenador de propósito general con un alto grado de especialización.

II.2.2.- Principio de lógica distribuida

Para eliminar la necesidad de mantener y procesar la información índice o estructural que en los sistemas convencionales permite simular por software el comportamiento asociativo de la memoria secundaria, hemos aplicado el principio de lógica distribuida como idea base en la organización especializada del PCBD. Se trata de dividir la memoria secundaria en bloques y asociar un elemento de procesamiento independiente a cada bloque (Figura II-2). El alto paralelismo de esta configuración permite la realización simultánea de la misma función de acceso sobre diferentes bloques de memoria, es decir, conseguir el comportamiento hardware asociativo de la memoria secundaria. Además, al distribuir la capacidad de procesamiento, los datos se procesan allí donde se encuentran, obviando la necesidad de su transmisión a un único procesador y, por tanto, disminuyendo el tiempo de respuesta de los accesos.

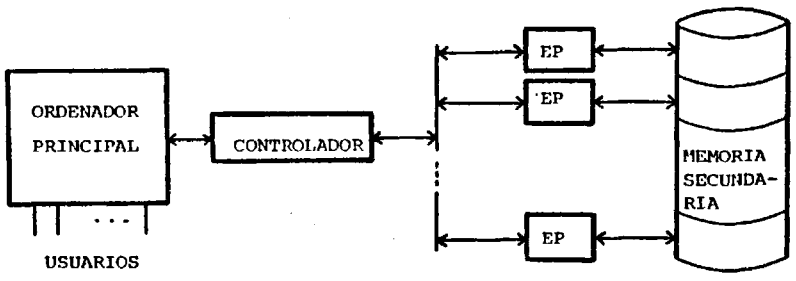


Figura II-2. Principio de lógica distribuida

Este principio fue utilizado primeramente por Slotnik ([7]) que asoció un elemento de lógica a cada pista de un dispositivo convencional de discos con una cabeza por pista (logic-per-track). Posteriormente su

uso se ha generalizado en virtud del abaratamiento de los componentes hardware. Ha sido empleado en los procesadores RAP ([8]), RARES ([9]), CASSM ([10]), RAPID ([11]) y DBC ([12]) con diferencias sustanciales que derivan fundamentalmente del modelo de datos utilizado, funciones asignadas a los elementos de procesamiento y capacidad de comunicación entre ellos.

II.2.3.- Elementos de procesamiento especializados

La complejidad de las funciones que los elementos de procesamiento realizan sobre los datos de sus respectivos bloques de memoria, así como el grado de rendimiento (medido en velocidad de proceso) alcanzado en su ejecución, determinarán la capacidad de procesamiento global de un dispositivo de lógica distribuida. En las diferentes alternativas que se han propuesto, la complejidad de estos elementos ha sido muy diversa. Así, mientras que en el prototipo de Slotnik se utilizaron simples dispositivos de lógica, en el procesador DIRECT ([13]) se han empleado microprocesadores bit-slice LSI-11/03 (PDP-11/03) con 28 K palabras de memoria para soportar los procedimientos que realizan las operaciones del álgebra relacional. Aunque los microprocesadores convencionales, debido a su flexibilidad de uso, resultarían adecuados para soportar funciones elementales de base de datos de diversa complejidad, su rendimiento, a causa de su arquitectura tipo Von Neumann, es muy bajo. En este punto, hay que tener muy en cuenta, además, la naturaleza de las memorias que actualmente y a plazo medio pueden utilizarse como soporte de una BD. Un análisis de las mismas revela una característica común: son memorias de naturaleza serie rotante, es decir, se comportan, desde el punto de vista lógico, como registros serie de desplaza-

miento cerrado a los que se accede por un único punto. Y para ello, basta citar los dispositivos magnéticos actualmente en uso, como son los tambores y discos, así como las memorias de reciente aparición MBM ([14]) y CCD ([15]).

Esta característica de los bloques de memoria exige en los elementos de procesamiento asociados una arquitectura adecuada, es decir, capaz de procesar un flujo serie de datos a la velocidad de rotación que la tecnología de la memoria imponga. Y es aquí donde los microprocesadores actuales presentan su mayor inconveniente.

Se han propuesto algunas arquitecturas para microprocesadores no-numéricos de tipo general. Se trata de procesadores "guiados" por datos (data driven) que ejecutan un conjunto de acciones genéricas, dentro del contexto del tratamiento no-numético, sobre memorias serie rotante ([16]). Sin embargo, al no considerar específicamente las exigencias de BD ni estar bien definidas sus funciones y pretenciones, su uso resulta bastante problemático, sobre todo al considerar el tema de la velocidad.

Todo lo anterior nos ha llevado a utilizar como elementos de proceso en el PCBD microprocesadores no-numéricos de propósito especial, es decir, que soportan directamente en su estructura un conjunto de operaciones (primitivas) adecuadas a las exigencias globales del procesador (figura II-3). De entre estas operaciones cabe destacar la búsqueda por contexto, responsable de la capacidad selectiva del procesador. Cada microprocesador irá asociado a un elemento de memoria rotante, formando ambos una célula. El primero, en el período de revolución del segundo, procesará todos sus datos en la forma especificada por la primitiva en ejecución.

Los procesadores CASSM ([19]) y RAP ([8]) también utilizan microprocesadores de propósito especial con características funcionales adecuadas

a sus respectivas concepciones.

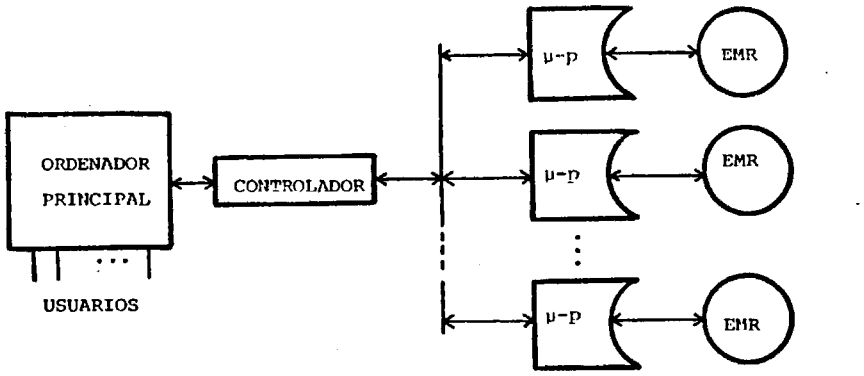


Figura II-3

II.2.4.- Implementación directa del modelo lógico

A fin de simplificar al máximo el mantenimiento y procesamiento de estructuras de datos auxiliares que en los SGBDs convencionales permiten establecer la correspondencia (mapping) entre modelo lógico de datos y representación física del mismo, hemos implementado directamente sobre las memorias rotantes el modelo relacional: modelo lógico del PCBD. A pesar de esta decisión de diseño, el PCBD podría utilizarse para un sistema no relacional (red o jerárquico) sin más que establecer la adecuada correspondencia en el ordenador principal.

En general, la representación de datos relativos a una entidad, en cualquier modelo lógico, exige tres tipos de información asociada que, con frecuencia, se almacenan sobre el mismo soporte de memoria secundaria:

- la relativa a la localización en la estructura lógica del modelo. En el jerárquico, por ejemplo, esta información define la posición nodal de los correspondientes datos.

- la descriptiva de su contenido y formato. Esto es, número de campos, longitud, etc.
- la relativa a la localización física sobre los soportes de memoria como punteros, índices, etc.

En el PCBD hemos eliminado la necesidad de almacenar sobre los elementos rotantes de memoria estos tres tipos de información. En efecto, la localización lógica de una entidad en el modelo relacional queda definida por la pertenencia del correspondiente tuple a una relación determinada. En el PCBD cada célula almacena tuples de una única relación, por lo que dicha información va implícita en la localización celular del tuple.

Al tener todos los tuples de una relación idéntico formato (número de dominios, longitud, etc.) la correspondiente información va asociada a los microprocesadores de las células que la soportan, obviando su almacenamiento repetitivo sobre los elementos de memoria.

Por último, los dispositivos de búsqueda por contexto de los microprocesadores hacen innecesario el almacenamiento de cualquier tipo de información relativa a la localización de tuples en una célula. Estos se almacenan en forma aleatoria sobre los elementos de memoria.

Esta tendencia a la representación directa del modelo lógico se observa en otros procesadores de BD propuestos. Así, CASSM utiliza un modelo jerárquico simple (de dos niveles) que le permite representar con facilidad los tres modelos lógicos utilizados en la actualidad: relacional, red y jerárquico. RAP utiliza el modelo relacional aunque manteniendo la información descriptiva de formato junto a los tuples almacenados.

II.2.5.- Capacidad de concurrencia

El comportamiento externo como memoria secundaria asociativa que el principio de lógica distribuida confiere a ciertos procesadores de BD propuestos ([17])([18]), elimina la necesidad de mantener y procesar todo tipo de información estructural (información referente a la localización de los datos). Se trata de procesadores celulares con una organización tipo SIMD (Single Instruction, Multiple Data), esto es, todas las células del procesador ejecutan la misma búsqueda en paralelo.

Un análisis de esta organización pone de manifiesto una importante limitación: no se aprovecha toda la capacidad de proceso que le confiere su alto paralelismo. En efecto, todas las células cuyos datos no pertenezcan al espacio de búsqueda referenciado por una función de acceso aportarán un proceso inútil a la ejecución de tal función.

Una utilización selectiva de las células permitiría la ejecución concurrente o simultánea de un conjunto de funciones de acceso con espacios de búsqueda disjuntos. Ello obligaría a mantener información de la distribución celular de los datos a nivel de la unidad encargada de gestionarlos. Sin embargo, teniendo en cuenta que otras exigencias de diseño, al obligar a una distribución controlada de datos sobre las células, simplifican la información relativa a su localización, resulta lógico su mantenimiento a cambio de un mejor aprovechamiento de los recursos de proceso totales.

Si a lo anterior unimos el hecho de que una BD integrada es un recurso a compartir por muchos usuarios, con exigencias simultáneas de acceder a ella, podemos concluir diciendo que una organización tipo MIMD (Multiple Instruction, Multiple Data) para los procesadores celulares de BD resulta más adecuada que la SIMD.

El PCBD, al organizarse como un procesador celular MIMD, podrá ejecutar concurrentemente un conjunto de accesos. Sin embargo, su capacidad de concurrencia no vendrá limitada por el solapamiento de los espacios totales de búsqueda de los accesos a ejecutar. Estos vendrán expresados en términos de una secuencia de instrucciones específicas ejecutables por el PCBD y referenciando, cada una de ellas, tan sólo un subconjunto de los datos totales del correspondiente acceso. Por consiguiente, el número de éstos que simultáneamente pueden ejecutar su instrucción pendiente vendrá condicionado solamente por el solapamiento de los subespacios de datos referenciados por dichas instrucciones.

La ocupación celular de las funciones de acceso se hará, pues, en forma controlada a medida que se ejecutan sus correspondientes instrucciones. De esta forma se posibilita una utilización multiplexada que aproveche al máximo la capacidad de procesamiento de todas las células.

II.3.- ORGANIZACION GENERAL DEL PCBD

El PCBD es un procesador semiautónomo concebido para soportar las funciones de gestión de BD de un ordenador de uso general ([19]). Es decir, se trata de un procesador "back-end" de propósito especial utilizado por un ordenador principal (host) para la realización de sus accesos a una BD. Su organización se especializa en torno al principio de lógica distribuida, esto es, un conjunto de microprocesadores especializados en la realización de funciones elementales de gestión sobre datos organizados según el modelo relacional y representados directamente sobre memorias de tipo rotante. Los accesos son ejecutados concurrentemente en virtud de una utilización conveniente del conjunto de microprocesadores.

II.3.1.- Relaciones con el ordenador principal

El ordenador principal soporta todas aquellas funciones de la base no relacionadas directamente con el acceso a los datos (figura II-4). Los usuarios acceden a la base a través del ordenador principal, bien desde un programa codificado con un lenguaje convencional como COBOL, PL/1, FORTRAN, etc., bien desde un terminal interactivo utilizando, en este caso, un lenguaje de interrogación (querry language) tal como SEQUEL, SQUARRE, etc. Los accesos, expresados en los diferentes lenguajes, son traducidos a programas codificados con instrucciones específicas de BD (instrucciones máquina del PCBD) y transmitidos al PCBD. Este, haciendo uso de la capacidad de procesamiento de su organización MIMD, los ejecuta concurrentemente y devuelve al área de trabajo de los respectivos usuarios, en memoria central del ordenador principal, los datos e información complementaria generados.

La capacidad de ejecución concurrente en el PCBD está limitada a un número N de programas (parámetros de diseño). Cada vez que uno de éstos finaliza, el ordenador principal es notificado de la disponibilidad de aquél para aceptar uno nuevo. Cuando el PCBD ha generado la información demandada por un acceso, la transmite al ordenador principal. Esta comunicación recíproca ORDENADOR PRINCIPAL ⟷ PCBD se realiza por Acceso Directo a Memoria (ADM) desencadenado por interrupción.

Las funciones de Integridad y Seguridad de la base no se soportan directamente en el PCBD, son mantenidas por el ordenador principal. Sin embargo, su gestión se ve fuertemente potenciada si tenemos en cuenta que las principales dificultades del establecimiento de tales funciones derivan del elevado número de accesos complementarios que exigen.

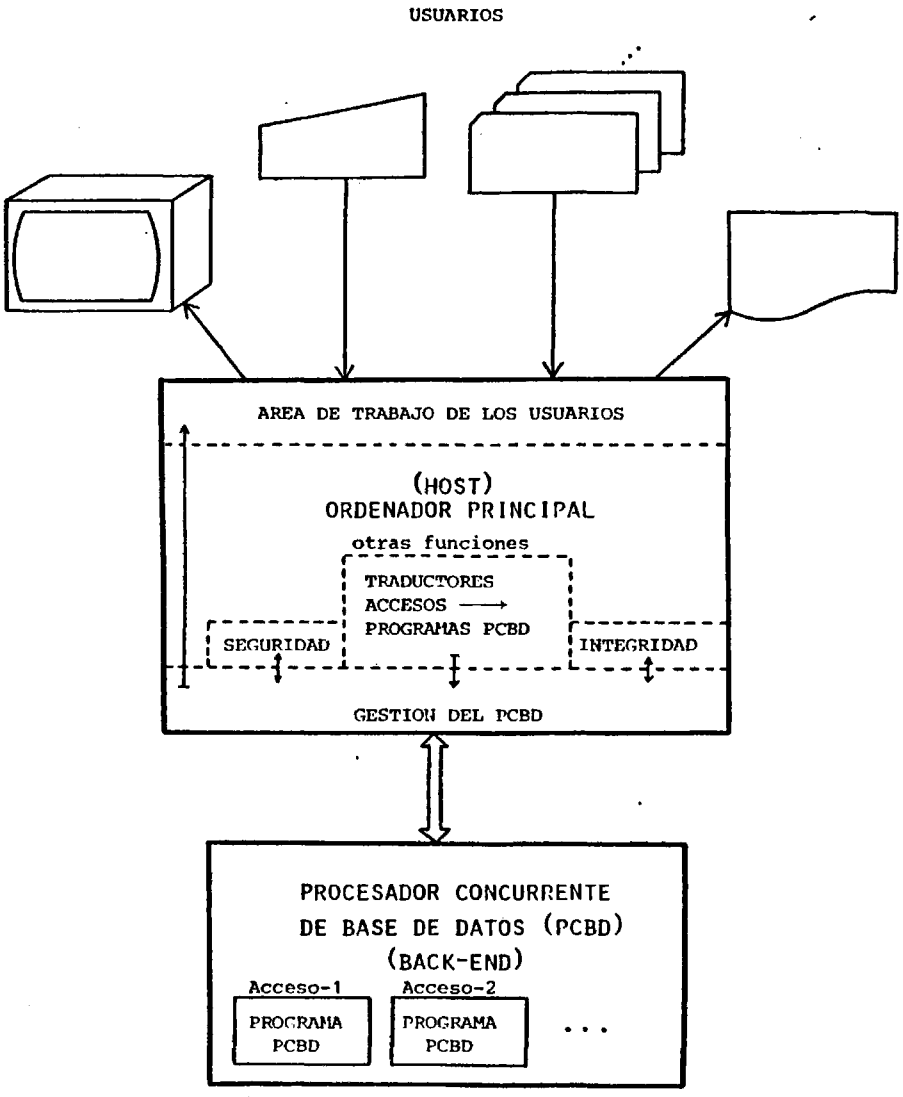


Figura II-4.- Relaciones entre el PCBD y el Ordenador Principal

II.3.2.- Estructura de bloques: funciones

El PCBD, como muestra la figura II-5, consta de dos componentes básicos: el Conjunto de Células (CC) y la Unidad de Coordinación (UC).

Cada célula está constituida por un Elemento de Procesamiento (EP) y un Elemento de Memoria (EM). Los EPs son microrpcoesadores no-numéricos de propósito especial que ejecutan operaciones elementales de BD sobre la información contenida en sus respectivos EMs. Estos son registros de desplazamiento circular serie con un único punto de acceso. El espacio total de memoria del PCBD, disponible para soportar los datos de la base, está constituido por la suma de todos los EMs. Las operaciones elementales ejecutadas por los EPs podemos clasificarlas en las típicas de inserción, recuperación, modificación y eliminación y se llevan a efecto en un período de revolución de los EMs bajo control de la UC. Cada célula funciona en modo solapado de inicialización/ejecución, es decir, durante una revolución, al mismo tiempo que ejecuta una operación elemental, es inicializada por la UC con la información necesaria a la operación elemental que habrá de ejecutar en la revolución siguiente. De esta forma se garantiza la disponibilidad de todas las células en cualquier revolución para que la UC le asigne la operación adecuada a la política de concurrencia establecida.

Las células pueden comunicarse entre sí, bajo control de la UC, a través de un bus bidireccional de intercambio (Figura II-5). En determinadas operaciones elementales, dos grupos de células utilizan esta facilidad para transmitirse datos directamente. El grupo emisor selecciona datos siguiendo un criterio de búsqueda y los transmite al grupo receptor para formar, en cada célula de éste, un nuevo criterio de búsqueda que aplicará a sus respectivos datos. Con este mecanismo, como estudiaremos en el capí-

tulo IV, se aceleran determinadas operaciones de BD del tipo n^2 y que en terminología relacional se traducen a la 0-composición.

Todas las células comparten un único bus unidireccional de salida (figura II-5) para transmitir los datos recuperados por este tipo de operaciones a la UC. El control de la utilización de este bus recae sobre la UC. Ambos buses, intercambio y salida, son asignados por la UC en función de la política de concurrencia establecida.

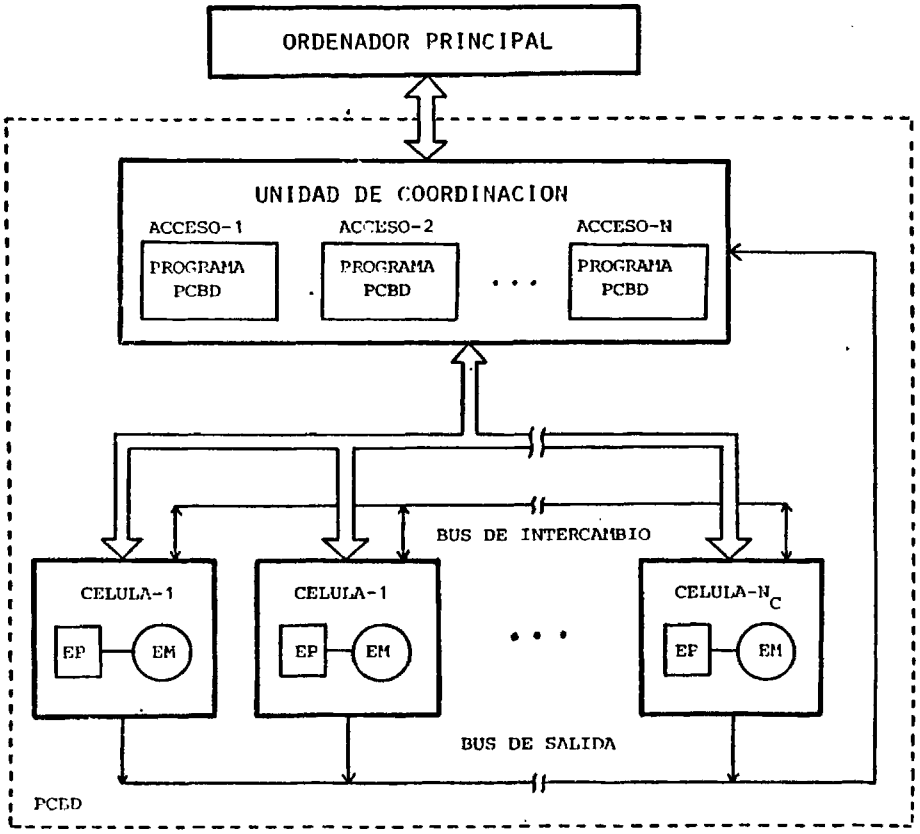


Figura II-5.- Estructura general de bloques del PCBD

La UC es la responsable de interactuar con el ordenador principal y controlar el CC, es decir, acepta programas PCBD correspondientes a los accesos de los usuarios, los ejecuta concurrentemente, sobre el CC, y devuelve al ordenador principal la información generada.

Para la ejecución de los programas PCBD, la UC decodifica sus respectivas instrucciones, determina los recursos de proceso (células y buses del CC) que cada una de ellas precisa, resuelve conflictos cuando más de una instrucción demanda un mismo recurso, distribuye las operaciones elementales pertinentes sobre el CC y recibe la información generada en la realización de las mismas. Todas estas funciones las realiza la UC en cada revolución síncrona de los elementos de memoria rotante del CC.

La resolución de los conflictos de utilización de un mismo recurso por diferentes programas PCBD se efectúa siguiendo una política de asignación de los mismos que tiene en cuenta el orden de prioridad de accesos establecido, el orden lógico de ejecución cuando intervienen accesos de modificación de la base y el aprovechamiento máximo de los recursos totales de proceso. En cualquier caso, el establecimiento de una política de asignación específica dependerá de las exigencias propias de cada sistema en particular y será por ello modificable (soportada por software).

II.3.3.- Representación de datos

Junto a las propiedades favorables que el modelo relacional de datos presenta respecto a las exigencias de los usuarios, su fácil linealización, lo convierten en un candidato idóneo para utilizarse como modelo de representación directa de datos sobre memorias de naturaleza rotante. En el PCBD hemos utilizado el modelo relacional con ligeras diferencias respecto

a la definición formal dada en el capítulo I. Estas diferencias no afectan a su concepción esencial y vienen impuestas por necesidades de diseño. Así, se permite la existencia de tuples redundantes en una relación, pero las instrucciones PCBD pueden controlar esta contingencia; se limita el grado de las relaciones (más precisamente, la longitud de los tuples) pero ello no es un óbice fundamental, ya que una relación siempre se puede descomponer en varias relaciones de menor grado, manteniéndose la totalidad de la información semántica; finalmente, se asocia a cada tuple un campo extra de información, accesible parte de él por los usuarios, y utilizado para el encadenamiento de operaciones.

En el PCBD los tuples de una relación se almacenan de manera desordenada sobre los elementos de memoria de las células. Cada tuple es una secuencia serie de bits, de longitud fija, dividida implícitamente en sus correspondientes dominios y un segmento de marca (figura II-6). Los dominios pueden ser discretamente variables de 1 a 4 bytes. El segmento de marca se compone de un bit de activación A y varios campos de marca M_i $(i=1,\ldots,N)$. El primero se utiliza para marcar los tuples eliminados de una relación por operaciones de borrado; sus posiciones pueden ser utilizadas para la inserción de tuples nuevos. El número N de campos M_i es un parámetro de diseño que determina el máximo número de programas PCBD que pueden ejecutarse concurrentemente. Cada función de acceso (programa PCBD) utiliza, durante su ejecución, un único campo M_i como área de trabajo temporal, a nivel de bits individuales, para marcar subconjuntos de tuples. De esta forma se posibilita que el resultado de una instrucción pueda utilizarse en instrucciones consecutivas. Cada campo M_i se compone de cuatro bits (m_0, m_1, m_2, m_3) permitiendo diferenciar 2^4 subconjuntos de tuples di-

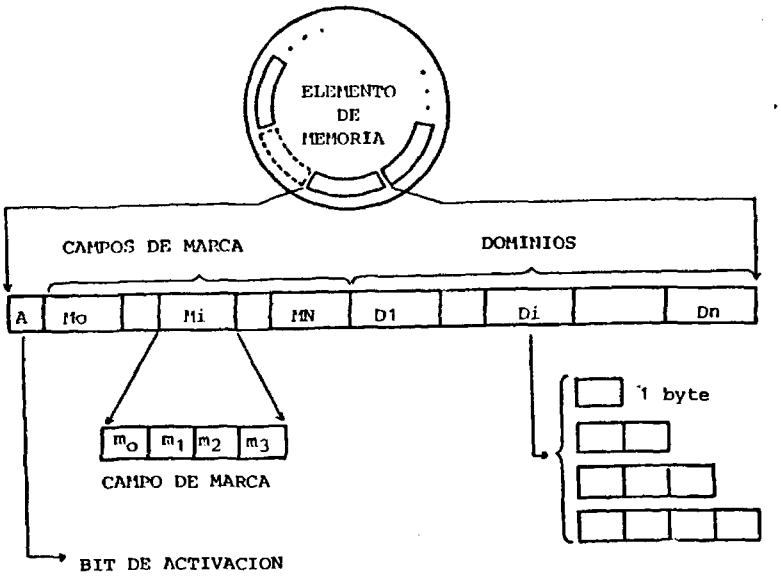


Figura II-6.- Formato y distribución de tuples en una célula

ferentes dentro de cada célula.

Cada elemento de memoria contiene tuples de una única relación, esto es, con formato idéntico. Cuando la capacidad de una célula no es suficiente para contener una relación completa, ésta se soporta por tantas células como sean necesarias. Siendo NC el número de células del PCBD, el número de relaciones posibles que puede soportar variará de 1 a NC, esto es, una única relación ocupando toda la capacidad de almacenamiento, o NC relaciones ocupando cada una de ellas una célula. La distribución de una relación entre diferentes células no impone tipo alguno de restricciones y es transparente a nivel de usuario. La referencia a una relación se hace, a nivel de usuario, a través de un nombre que la iden-

tifica unívocamente sin necesidad de especificar su localización celular. Sin embargo, como veremos más adelante, las instrucciones PCBD posibilitan el control de datos a nivel de células individuales. Esta decisión de diseño obedece a dos razones principales. En primer lugar, este control es necesario en la fase de creación de la base. En segundo lugar, permite un mejor aprovechamiento de la capacidad total de almacenamiento cuando, después de frecuentes operaciones de borrado, es posible reagrupar la información válida de una relación en un menor número de células, ampliándose el espacio de almacenamiento libre de la base.

II.4.- ARQUITECTURA GENERAL DEL PCBD

La estructura conceptual y comportamiento funcional del PCBD ([20]), es decir, su arquitectura general, vendrá definida por el repertorio de instrucciones que ejecuta y las estructuras de datos que soporta, tal y como son vistas desde el ordenador principal. Estudiaremos, pues, en este apartado la vista relacional de los datos y forma de acceder a ellos independientemente del procesamiento concurrente que tiene lugar durante su ejecución. De este último aspecto nos ocuparemos en el apartado siguiente.

Un acceso será un programa construido con instrucciones máquina (instrucciones PCBD). Estos harán referencia a las relaciones de la base. Una relación es una estructura tabular normalizada (Fig. II-7) con un nombre que la identifica. El número de filas (tuples) que la componen no es significativo puesto que las instrucciones operan sobre relaciones completas (operaciones orientadas a conjuntos). Cada columna (dominio) de una relación se identifica con el correspondiente nombre (D_i). La primera de todas, denominada columna o campo de marca (M), se utilizará,

como vimos en el apartado II.3.3, para el encadenamiento de operaciones.

NOMBRE DE RELACION	M				D_1	...	D_i	...	D_n
	m_0	m_1	m_2	m_3					
	1	1	0	1	d_{11}		d_{i1}		d_{n1}
	0	1	1	0	d_{12}		d_{i2}		d_{n2}
	.....				...	...	...	...	...

Figura II-7

En la expresión de un procedimiento (programa PCBD) que accede a la base, se hará referencia, además de a las relaciones, a un conjunto de registros r_i y a un Area de Entrada/Salida (A.E/S) (Figura II-8). Los primeros localizados en el PCBD, se utilizarán como área de trabajo temporal. Su número será igual al número máximo de dominios (grado) permitidos en una relación (parámetro de diseño). Cada registro tendrá capacidad para almacenar un valor de un dominio de un tuple. A través del A.E/S, localizada en la memoria central del ordenador principal, se realizará el intercambio de información (transmisión y recepción) entre la base y el usuario.

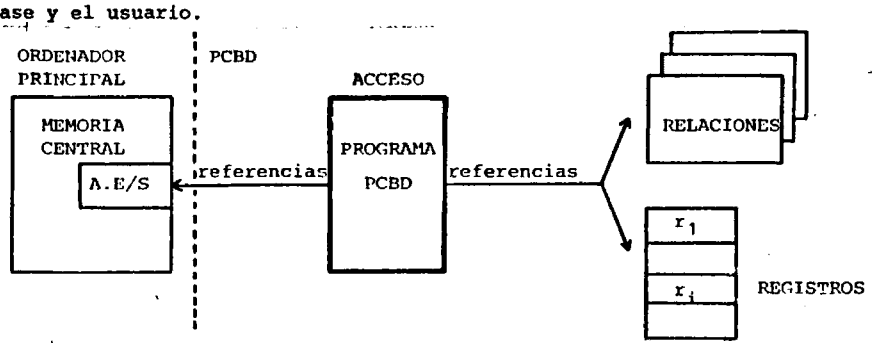


Figura II-8

II.4.1.- Repertorio de instrucciones

En la fijación de las instrucciones ejecutables por el PCBD se han tenido en cuenta los siguientes compromisos:

1. Conseguir un repertorio relacionalmente completo, es decir, que permita la codificación de cualquier acceso expresable en cálculo relacional. Como se vio en el capítulo I, es condición suficiente para que un sublenguaje de datos sea relacionalmente completo que en él sean expresables las ocho operaciones del álgebra relacional. En el apartado II.4.1.4. hemos demostrado el carácter completo del repertorio PCBD.

2. Proximidad semántica a los sublenguajes de datos de alto nivel. De esta forma se simplifican los procesos de traducción de este tipo de lenguajes. Como veremos en el apartado II.4.1.3, la mayoría de las instrucciones constituyen expresiones relacionales de alto nivel con una gran capacidad selectiva.

3. Minimizar el tiempo de ejecución de las operaciones de BD de tipo n^2, esto es, las que implican la comparación de dos conjuntos de n datos y que en terminología relacional equivalen, esencialmente, a la θ-composición. En el capítulo IV se hace un estudio cuantitativo, en forma comparada, de esta característica del repertorio.

4. Posibilidad de su implementación hardware en una organización celular, como es la del PCBD, permitiendo su ejecución concurrente y multiplexada para hacer posible el control de los recursos del proceso. Esto equivale a la descomponibilidad de cada instrucción en una sucesión de comandos de célula con el mismo tiempo de ejecución (tiempo de revolución de la memoria rotante).

II.4.1.1.- Formato de las instrucciones

En general, una instrucción PCBD especifica una acción a realizar sobre los tuples de una relación que cumplan determinadas condiciones. Su formato general es:

$[CO]\,[R]\,[DM]\,[C]\,[P.E.]$ siendo:

$[CO]$ Código de Operación, especifica la acción a ejecutar

$[R]$ nombre de la Relación, espacio de datos al que afecta

$[DM]$ Descripción de Marca, especifica el nuevo valor del campo de marcas que tomarán los tuples de R que cumplan determinadas condiciones

$[C]$ Cualificación o conjunto de condiciones exigidas a los tuples de R para que sobre ellos se efectúen las acciones especificadas en $[CO]$ y $[DM]$

$[P.E.]$ conjunto de Parámetros Especiales cuya función es específica para cada instrucción.

Cuando la acción de la instrucción afecta únicamente a un subconjunto de dominios de la relación, el nombre R, va seguido por el nombre de estos dominios entre paréntesis: $R(D_1, D_2, \ldots)$. Cuando la acción se restringe a un subconjunto de células de las que soportan la relación, las direcciones de éstas serán las que seguirán al nombre R: $R(c_1, c_2, \ldots)$.

La nueva asignación de valores a los bits de marca la especificaremos con la presencia, complementada o no, de las variables binarias que identifican las posiciones del campo en la Descripción de Marca. La variable complementada asignará el valor binario cero; no complementada uno; y si no aparece, su valor no variará. Así, la descripción de marca $[m_0\ \overline{m_1}\ m_3]$ hará, para todos los tuples que cumplan la cualificación, $m_0=1$, $m_1=0$, $m_3=1$ y m_2 no se modificará.

Una cualificación sera, para todas las instrucciones a excepción de la Selección Indirecta, una expresión booleana conjuntiva o disyuntiva simple de términos T_i:

$$C = \bigwedge_{i=1}^{n} T_i \qquad o \qquad C = \bigvee_{i=1}^{n} T_i \qquad (n \leq \text{grado de } R + 1)$$

Cada término T_i puede ser de dos tipos:

- tipo dato. Es un triplete de la forma: (DOMINIO θ COMPARANDO), donde DOMINIO es uno cualquiera de la relación referenciada; $\theta \in \{<,\leq,=,\geq,>\}$ y COMPARANDO puese ser: el nombre de otro dominio diferente de la relación o una constante. En este último caso o bien explícitamente declarada en la cualificación o bien contenida en un registro r_i de la UC. En una cualificación pueden aparecer referenciados, como dominio o comparando, pero una sola vez, todos los dominios de la relación.
- tipo marca. Es una expresión booleana conjuntiva o disyuntiva simple de términos t_i $(i=1,\ldots,4)$:

$$T_m = \bigwedge_{i=1}^{4} t_i \qquad o \qquad T_m = \bigvee_{i=1}^{4} t_i$$

Cada término t_i es de la forma: $(m_i = v.b.)$ donde m_i es una variable que identifica una posición de marca y $v.b. \in \{0,1\}$. Por simplicidad de escritura, los términos conjuntivos de tipo marca los escribiremos en la forma M=configuración de cuatro bits. Los "1" de la misma indicarán términos $(m_i = 1)$, los "0" términos $(m_i = 0)$ y cuando el correspondiente término no aparece, lo denotaremos con "x" en la posición correspondiente. Así, el término tipo marca: $(m_1 = 1) \wedge (m_3 = 0)$ lo escribiremos: $M = 1 x x 0$. Análogamente haremos para los términos tipo marca disyuntivos,

precediendo, en este caso, la configuración de bits con la letra "d". Así, el término $(m_1 = 1) \vee (m_3 = 0)$ lo escribiremos: $M = d1xx0$. En una cualificación sólo puede aparecer un término tipo marca.

Como ejemplo de cualificación sobre la relación R(A, B, C, D) tenemos:

$$C \equiv \underbrace{[(M = 00x1)}_{\text{término de marca}} \wedge \underbrace{(A > 3) \wedge (B = C) \wedge (D = n_1)]}_{\text{términos de datos}}$$

Teniendo en cuenta que cualquier expresión booleana, por compleja que sea, es siempre expresable en forma conjuntiva o disyuntiva normal, su resolución será siempre posible en el PCBD sin más que ejecutar una secuencia de operaciones de selección con cualificaciones disyuntivas o conjuntivas simples encadenadas a través del campo de marcas.

Una característica a destacar en las cualificaciones del PCBD la constituye la posibilidad de expresar criterios de búsqueda por contexto. En efecto, este tipo de búsqueda selectiva supone una generalización de la simple búsqueda por contenido implementada en otros procesadores de BD [21][22]. Consiste en la selección de tuples de una relación en función de una comparación realizada sobre dos de sus dominios. En definitiva y como veremos en el apartado II.4.1.4, se trata de la implementación directa de la restricción relacional. Para poner brevemente de manifiesto la ventaja de la búsqueda por contexto, supongamos que sobre la relación EMPLEADO (NOMBRE, SALARIO, COMISION) se formula el siguiente acceso: seleccionar todos los empleados que ganan en concepto de comisión más que por su salario. En el PCBD bastaría con ejecutar una selección con la cualificación (COMISION > SALARIO) que se realizaría en

una revolución de la memoria rotante. En otros procesadores celulares que sólo dispongan de búsqueda por contenido ([23]) habría que seleccionar en una revolución el valor de salario de un tuple y en otra revolución compararlo con el de comisión, y esto para todos los tuples de la relación.

II.4.1.2.- Descripción de los diferentes grupos de instrucciones

Cada instrucción del repertorio la estudiaremos dentro de un tipo o grupo, es decir, conjunto de instrucciones que responden a una función general de BD (definición, recuperación, modificación), o a un aspecto relacionado con el rendimiento o con la proximidad a los lenguajes de alto nivel.

(1) Instrucciones de Definición. Las dos instrucciones que componen este grupo permiten la definición inicial de la base y su posible reestructuración. No existen en ellos ni cualificación ni descripción de marça.

CREAR (CR)

formato: [CR] [R $(c_1,\ldots,c_n)$] [*longitud de la relación*]

función: asigna las células de direcciones $c_1,\ldots,c_n$ a la relación de nombre R y declara la longitud de sus tuples así como la de sus dominios componentes.

ELIMINAR (EL)

formato: [EL] [R $(c_1,\ldots,c_n)$]

función: libera a las células $c_1,\ldots,c_n$ de la relación R, quedando a disposición de su asignación a una nueva relación.

(2) Instrucciones básicas de recuperación. Componen este grupo las cua-

tro instrucciones que garantizan el carácter completo del repertorio, es decir, con ellas se pueden codificar programas para las ocho operaciones del álgebra relacional.

SELECCION DIRECTA (SD)

formato: $[SD]$ $[R]$ $[DM]$ $[C]$

función: selecciona todos los tuples de la relación R que cumplen la cualificación C. La selección consiste en modificar el campo de marca según DM.

ejemplo: $[SD]$ $[R]$ $[m_0]$ $[(M=0xx0) \wedge (A=a) \wedge (B > 5)]$

	M	A	B
R	0 0 0 0	a	6
	0 1 0 0	b	5
	0 0 0 0	a	7

(antes de la ejecución)

	M	A	B
R	1 0 0 0	a	6
	0 1 0 0	b	5
	1 0 0 0	a	7

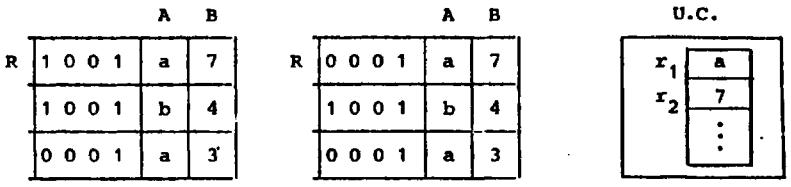
(después de la ejecución)

RECUPERACION INTERNA (RI)

formato: $[RI]$ $[R(D_1,\ldots,D_n)]$ $[DM]$ $[C]$ $[r_1,\ldots,r_n]$

función: los dominios $D_1,\ldots,D_n$ de un solo tuple de R que cumpla la cualificación C se transmiten a los registros $r_1,\ldots,r_n$ de la UC. Además, este tuple se marca según DM.

ejemplo: $[RI]$ $[R(A,B)]$ $[\overline{m_0}]$ $[(M=1001) \wedge (A=a) \wedge (B > 11)]$ $[r_1,r_2]$



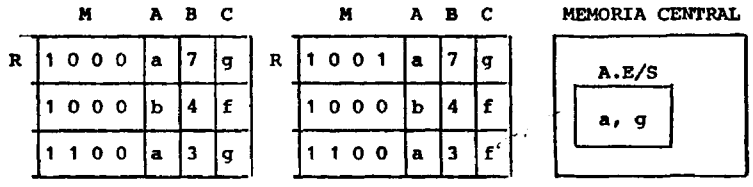
(antes de la ejecución) (después de la ejecución)

RECUPERACION EXTERNA (RE)

formato: [RE] [$R(\mathcal{D}_1,\ldots,\mathcal{D}_n)$] [$\mathcal{D}'$] [C] [A.E/S]

función: los dominios $\mathcal{D}_1,\ldots,\mathcal{D}_n$ de los tuples de R que cumplen la cualificación C se transmiten al Area de E/S en memoria central del ordenador principal. Al mismo tiempo estos tuples se marcan según $\mathcal{DM}$.

ejemplo: [RE] [R(A,C)] [m_3] [$(M=1x0x) \wedge (A=a) \wedge (B > 2) \wedge (C=g)$]



(antes de la ejecución) (después de la ejecución)

BIFURCACION CONDICIONAL (BC)

formato: [BC] [R] [C] [$\mathcal{DS}$]

función: transfiere control a la instrucción de dirección $\mathcal{DS}$ cuando algún tuple de R cumple la cualificación C. En caso contrario, la ejecución del programa sigue el orden secuencial. Esta instrucción permite las iteraciones controladas necesarias en la rea-

lización de proyecciones o composiciones físicas de relaciones, así como en actualizaciones que implican más de una operación aritmética.

(3) Instrucciones de actualización. Componen este grupo las instrucciones que hacen posible la actualización del contenido de la base, esto es, la inserción, borrado y modificación de tuples. En general, una modificación ha de descomponerse en la siguiente secuencia:

recuperación de los tuples a modificar
borrado de los tuples a modificar
modificación fuera del PCBD
inserción de los tuples modificados

Sin embargo, hemos incluido en el repertorio, además de las instrucciones de borrado e inserción, tres instrucciones que permiten modificaciones simples sobre un dominio de una relación de forma directa. Estas modificaciones consisten en sustituir, añadir y restar sobre el citado dominio.

BORRADO (BO)

Formato: [*BO*] [*R*] [*C*]

función: desactiva (pone a cero) el bit A de todos los tuples de R que cumplan la cualificación C.

INSERCION (IN)

formato: [*IN*] [*R*] [*A.E/S*]

función: introduce sobre posiciones de tuples libres (bit A desactivado) de la relación *R* los tuples contenidos en el Area de E/S del usuario, en memoria central del ordenador principal.

MODIFICACION SIMPLE (SU, RS, SS)

formato: $\left\{\begin{array}{l}[SU]\\ [RS]\\ [SS]\end{array}\right\}$ $[R(D)]$ $[DM]$ $[C]$ $[OP]$

función: todos los valores del dominio D, pertenecientes a tuples que cumplan la cualificación C, son modificados (sumados, restados o sustituidos) con el operando OP. Este puede ser una constante explícitada, el contenido de un registro r_i, o el valor de otro dominio de R (modificación por contexto).

Ejemplos: partiendo de la relación:

	M	A	B	C
R	0 0 1 0	a	7	4
	1 1 1 1	a	3	5

veamos las sucesivas modificaciones originadas por las instrucciones siguientes.

$[SU]$ $[R(B)]$ $[m_0\ \overline{m_1}]$ $[(M=xx1x) \wedge (A=a)]$ $[8]$

	M	A	B	C
R	1 0 1 0	a	15	4
	1 0 1 1	a	11	5

$[RS]$ $[R(C)]$ $[m_3]$ $[(M=10xx)]$ $[2]$

	M	A	B	C
R	1 0 1 1	a	15	2
	1 0 1 1	a	11	3

$[SS]\ [R(B)]\ [\overline{m_3}]\ [(B > 10) \wedge (C < 11)]\ [C]$

	M	A	B	C
R	1 0 1 0	a	2	2
	1 0 1 0	a	3	3

(4) Instrucciones de Agregación. Este grupo lo constituyen instrucciones que permiten calcular directamente funciones de agregación, tales como el valor máximo, mínimo, promedio, total o número. Su uso simplifica la traducción de expresiones relacionales de sublenguajes de datos, tales como ALPHA, SEQUEL, etc., que permiten este tipo de funciones.

formato: $\left\{\begin{matrix}[MA]\\[MI]\\[PR]\\[TO]\\[NU]\end{matrix}\right\}\ [R(D)]\ [DM]\ [C]\ [r_i]$

función: sobre todos los valores del dominio D, pertenecientes a tuples que cumplan la cualificación C, se calcula el máximo, mínimo, promedio, total o número respectivamente, y se almacena en el registro r_i. Los tuples cualificados se marcan según DM. Para la instrucción NU no se especifica D.

ejemplos:

	M	A	B	C
R	0 0 0 0	a	5	1
	0 0 0 0	b	4	3
	0 0 0 0	a	3	9
	0 0 0 0	a	8	4
	0 0 0 0	d	1	3

				resultados r_1
[MA]	[R(B)]	[ø]	[(A=a) ∧ (B < 9)] [r_1]	8
[MI]	[R(C)]	[ø]	[(A=a) ∨ (B > 1)] [r_1]	1
[PR]	[R(C)]	[ø]	[(A=a) ∨ (B=1)] [r_1]	4
[TO]	[R(C)]	[ø]	[(A=a) ∨ (B=1)] [r_1]	20
[NU]	[R]	[ø]	[(A=a) ∨ (B=1)] [r_1]	5

[ø] significa que no se actúa sobre el campo de marca.

(5) Selección Indirecta (SI). Se trata de una instrucción que, aunque no introduce nuevos recursos selectivos, acelera los procesos de búsqueda que implican la comparación de dos conjuntos grandes de datos (tipo n^2). Realiza la composición implícita de dos relaciones, es decir, selecciona tuples de una relación según un criterio de búsqueda formado con datos seleccionados en otra y transmitidos directamente a su células soporte por el bus de intercambio.

formato: [SI] [R1] [DM1] $\{(D_1 \ \theta \ D2)$ [R2] [DM2] [C]$\}$

función: selecciona, modificando el campo de marca según DM1, los tuples de la relación R1 cuyo dominio D1 cumpla la comparación $\theta \in \{<, \leq, =, \geq, >\}$ con cualquier valor del dominio D2 de los tuples de R2 que cumplan la cualificación C. Además, los tuples de R2 que cumplen C se modifican según DM2.

ejemplo:

[SI] [R] [m_0] {(A=Z) [S] [$\overline{m_0}$] [(x > 5) ∧ (Z=a)]}

R

M	A	B
0 0 0 0	b	f
0 0 0 0	a	g
0 0 0 0	a	g
0 0 0 0	a	h

S

M	X	Z
1 0 0 0	6	a
1 0 0 0	5	a
1 0 0 0	4	b
1 0 0 0	7	c

(antes de la ejecución)

R

M	A	B
0 0 0 0	b	f
1 0 0 0	a	g
1 0 0 0	a	g
1 0 0 0	a	b

S

M	X	Z
0 0 0 0	6	a
1 0 0 0	5	a
1 0 0 0	4	b
1 0 0 0	7	c

(después de la ejecución)

(6) Instrucciones de control. Componen este grupo dos instrucciones que permiten controlar los espacios libres de la memoria rotante.

ESPACIO (ES)

Formato: $[ES]\ [R(c_1,\ldots,c_n)]\ [x_i]$

Función: calcula el número de posiciones de tuple libres en las células $c_1,\ldots,c_n$ de la relación R y lo almacena en x_i.

EMPAQUETAR (EM)

Formato: $[EM]\ [R(c_1,\ldots,c_n)]$

Función: coloca los tuples activos de las células $c_1,\ldots,c_n$ de R en las primeras posiciones de sus respectivos elementos de memoria.

(7) Otras instrucciones. Agrupamos aquí un conjunto de instrucciones cuya ejecución no implica la utilización del CC, es decir, se resuelven a

nivel de la U.C. Su función es la de proporcionar flexibilidad en la codificación de los accesos y definir su finalización.

CARGAR (CA)

Formato: [CA] [$r_1,\ldots,r_n$] [$v_1,\ldots,v_n$]

Función: carga los registros $r_1,\ldots,r_n$ con los valores $v_1,\ldots,v_n$. Estos pueden ser explícitamente declarados o estar contenidos en otros registros.

ALMACENAR (AM)

Formato: [AM] [$r_1,\ldots,r_n$] [A.E/S]

Función: lleva directamente el contenido de los registros $r_1,\ldots,r_n$ al Area E/S del usuario.

SALTO CONDICIONAL (SC)

Formato: [SC] [$r_i \ \theta \ r_j$] [DS]

Función: transfiere control a la instrucción situada en DS si la condición $r_i \ \theta \ r_j$ es cierta. En caso contrario sigue en secuencia.

ALTO (AL)

Formato: [AL]

Función: finaliza la codificación de un programa PCBD.

II.4.1.3.- Utilización del repertorio

Para poner de manifiesto la expresión de accesos en términos del repertorio PCBD, veamos algunos ejemplos sobre la base definida en el apartado I.3 y que reproducimos a continuación:

PILOTO (PI#, PINOM, CR)

AVION (AV#, AVNOM, CAP, CB)

VUELO (VU#, PI#, AV#, CO, CD, HP, HL)

A1. Recuperar los números de los pilotos (PI#) que realizan vuelos entre Madrid y Sevilla.

[RE] [VUELO (PI#)] [ø] [(CO = MADRID) ∧ (CD= SEVILLA)] [A.E/S]

* Lleva a A.E/S los números de los pilotos que realizan vuelos partiendo de Madrid con destino Sevilla

[RE] [VUELO (PI#)] [ø] [(CO = SEVILLA) ∧ (CD = MADRID)] [A.E/S]

* Lleva a A.E/S los números de los pilotos que realizan vuelos partiendo de Sevilla con destino Madrid

[AL]

A2. Recuperar el nombre del avión con base en Madrid que tenga mayor capacidad.

[MA] [AVION (CAP)] [ø] [(CB = MADRID)] [r_1]

* Lleva a r_1 el máximo valor de capacidad de todos los aviones

[RE] [AVION (AVNOM)] [ø] [(CAP = <r_1>)] [A.E/S]

[AL]

A3. Recuperar los nombres de los pilotos que realizan vuelos entre Madrid y Sevilla.

[SD] [PILOTO] [$\overline{m_o}$] [ø]

[SD] [VUELO] [$\overline{m_o}$] [ø]

* Borran la marca m_o en las dos relaciones

[SD] [VUELO] [m_o] [(CO = MADRID) ∧ (CD = SEVILLA)]

* Marca m_o los tuples de VUELO que parten de Madrid con destino Sevilla

[SD] [VUELO] [m_o] [(CO = SEVILLA) ∧ (CD = MADRID)]

* Marca m_o los tuples de VUELO que parten de Sevilla con destino Madrid

[SI] [PILOTO] [m_o] {(PI# = PI#) [VUELO] [Ø] [(M = 1xxx)]}

* Marca m_o en PILOTO todos los tuples con PI# igual a PI# de cualquier tuple de VUELO marcado m_o

[RE] [PILOTO (PINOM)] [$\overline{m_o}$] [(M = 1xxx)] [A.E/S]
[AL]

A4. Retrasar en una hora la partida y llegada de todos los vuelos conducidos por los pilotos residentes en Madrid.

[SD] [PILOTO] [$\overline{m_o}$] [Ø]
[SD] [VUELO] [$\overline{m_o}$] [Ø]
[SD] [PILOTO] [m_o] [(CR = MADRID)]
[SI] [VUELO] [m_o] {(PI# = PI#) [PILOTO] [Ø] [M = 1xxx] }
[RS] [VUELO (HP)] [$\overline{m_o}$] [(M = 1xxx)] [1]
[RS] [VUELO (HL)] [$\overline{m_o}$] [(M = 1xxx)] [1]
[AL]

A5. Recuperar el nombre de todos los pilotos residentes en Sevilla que realizan algún vuelo con destino Madrid.

Para la expresión de este acceso utilizaremos la siguiente codificación en el campo de marcas de la relación PILOTO

m_o	m_1	
0	0	tuples no procesados
0	1	tuple en fase de procesamiento
1	0	tuples procesados y seleccionados
1	1	tuples procesados y no seleccionados

[SD] [PILOTO] [$\overline{m_o}$ $\overline{m_1}$] [Ø]

* Todos los tuples de PILOTO pasan al estado de no procesados

[BC] [PILOTO] [(CR = SEVILLA)] [1]

* Bifurca a ALTO si no existen pilotos residentes en Sevilla

[BC] [ø] [ø] [4]

1 [RI] [PILOTO (PI#)] [m_1] [(M = 00xx) ∧ (CR = SEVILLA)] [r_1]

* Un tuple de PILOTO con residencia en Sevilla pasa a la fase de procesamiento almacenándose en r_1 un dominio PI#

[BC] [VUELO] [(PI# = <r_1>) ∧ (CD = MADRID)] [2]

* Prueba y bifurca a [2] si el tuple de PILOTO en fase de procesamiento cumple la condición de realizar algún vuelo con destino MADRID

[SD] [PILOTO] [m_o m_1] [(M = 01xx)]

* En caso de no cumplir la condición, el tuple en fase de procesamiento pasa a estado de procesado y no seleccionado.

[BC] [ø] [ø] [3]

* Bifurca incondicionalmente a [3]

2 [SD] [PILOTO] [m_o $\overline{m_1}$] [M = 01xx)]

* El tuple en fase de procesamiento pasa al estado de procesado y seleccionado

3 [BC] [PILOTO] [M = 00xx) ∧ (CR = SEVILLA)] [1]

* Bifurca a [1] si existen en PILOTO más tuples no procesados

[RE] [PILOTO (PINOM)] [ø] [(M = 10xx)] [A.E/S]

* En caso contrario lleva al A.E/S todos los tuples procesados y selec-

cionados

4 [AL]

II.4.1.4.- Potencia selectiva del repertorio

Demostraremos aquí el carácter completo del repertorio PCBD, es decir, que dispone, al menos, de la potencia selectiva del álgebra y cálculo relacional. Para ello, sólo utilizaremos las Instrucciones Básicas de Recuperación.

La expresión de las operaciones del álgebra relacional en términos del repertorio PCBD, que veremos seguidamente, sólo tiene un interés teórico: demostrar la "completitud" relacional del mismo tal como fue definida por Codd. La formulación de cualquier acceso, como acabamos de ver en la sección precedente, se puede realizar directamente, sin necesidad de pasar por su expresión algebraica relacional.

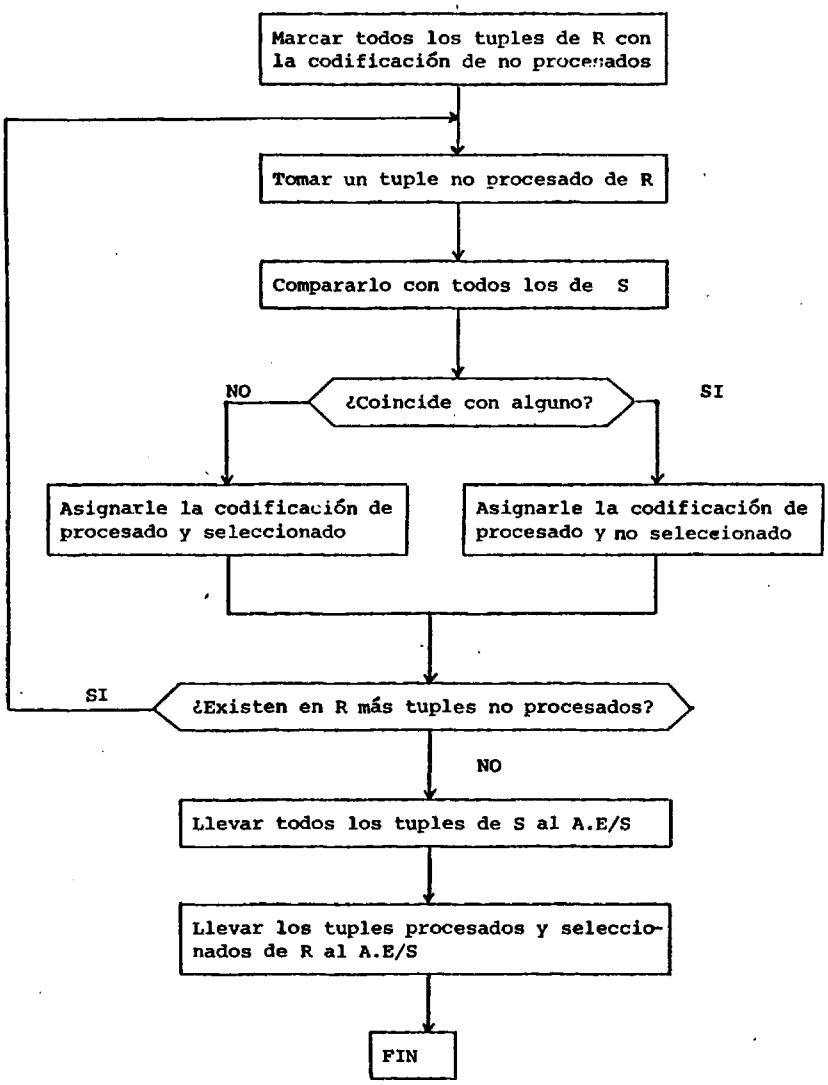
- UNION: $R(D_1,\ldots,D_n) \cup S(D_1,\ldots,D_n)$

La unión de estas dos relaciones estará formado por todos los tuples de S, y los de R no contenidos en S. En la codificación del programa PCBD correspondiente supondremos que S y R no contiene tuples redundantes. En caso contrario, bastaría realizar previamente una proyección sobre sí misma de acuerdo al programa de esta operación relacional que veremos más adelante.

Codificación de marcas en R:

m_0	m_1	
0	0	tuples no procesados
0	1	tuple en fase de procesamiento
1	0	tuples procesados y seleccionados
1	1	tuples procesados y no seleccionados

Organigrama:



Programa PCBD:

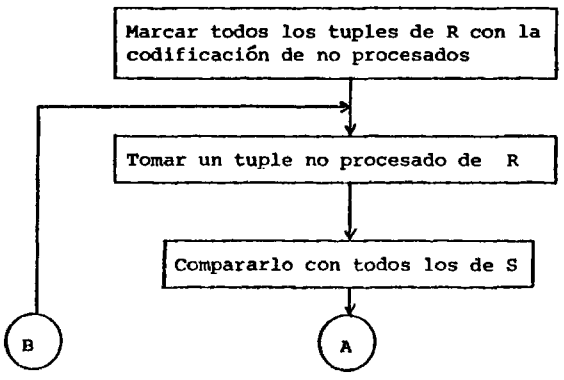
	CO	RELACION	DM	CUALIFICACION	P.E.
	[SD]	[R]	$[\overline{m_o}\ \overline{m_1}]$	[ø]	
1	[RI]	$[R(D_1,\ldots,D_n)]$	$[\overline{m_o}\ m_1]$	[(M=00xx)]	$[r_1,\ldots,r_n]$
	[BC]	[S]		$[(D_1=<r_1>)\wedge\ldots\wedge(D_n=<r_n>)]$	[2]
	[SD]	[R]	$[m_o\ \overline{m_1}]$	[(M=01xx)]	
	[BC]	[ø]		[ø]	[3]
2	[SD]	[R]	$[m_o\ m_1]$	[(M=01xx)]	
3	[BC]	[R]		[(M=00xx)]	[1]
	[RE]	$[S(D_1,\ldots,D_n)]$	[ø]	[ø]	[A.E/S]
	[RE]	$[R(D_1,\ldots,D_n)]$	[ø]	[(M=10xx)]	[A.E/S]
	[AL]				

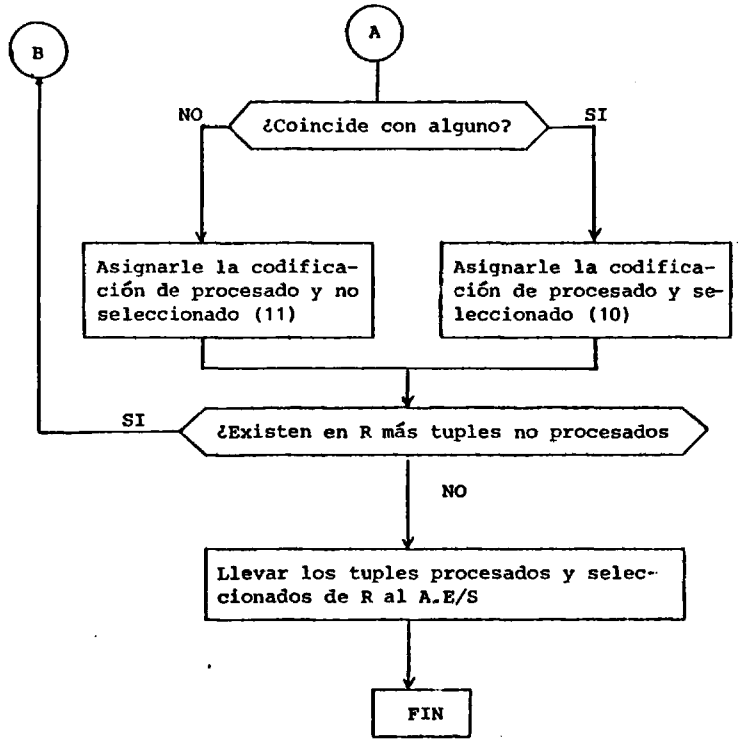
- INTERSECCION: $R(D_1,\ldots,D_n) \cap S(D_1,\ldots,D_n)$

Lo mismo que en la Unión, supondremos que en S y R no existen tuples redundantes.

Codificación de marcas en R: lo mismo que en el programa de la Unión.

Organigrama:





Programa PCBD:

	CO	RELACION	DM	CUALIFICACION	P.E.
	[SD]	[R]	$[\overline{m_o}\ \overline{m_1}]$	[ø]	
1	[RI]	$[R(D_1,\ldots,D_n)]$	$[\overline{m_o}\ m_1]$	[(M=00xx)]	$[r_1,\ldots,r_n]$
	[BC]	[S]		$[(D_1=<r_1>)\wedge\ldots\wedge(D_n=<r_n>)]$	[2]
	[SD]	[R]	$[\overline{m_o}\ \overline{m_1}]$	[M=01xx)]	
	[BC]	[ø]		[ø]	[3]
2	[SD]	[R]	$[m_o\ \overline{m_1}]$	[(M=01xx)]	
3	[BC]	[R]		[(M=00xx)]	[1]
	[RE]	$[R(D_1,\ldots,D_n)]$	[ø]	[M=10xx)]	[A.E/S]
	|AL|				

- DIFERENCIA: $R(D_1,\ldots,D_n) - S(D_1,\ldots,D_n)$

Teniendo en cuenta la definición de diferencia, el programa PCBD correspondiente es idéntico al de la Unión, sin más que eliminar de éste la instrucción de Recuperación Externa que lleva al A.E/S todos los tuples de S (instrucción octava del programa).

- PRODUCTO CARTESIANO: $R(D_1,\ldots,D_n) \oplus S(D'_1,\ldots,D'_n)$

Para la realización de esta operación habrá que llevar al A.E/S la concatenación de cada tuple de R con cada uno de los de S.

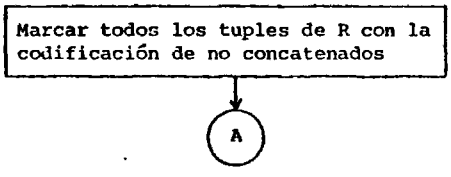
Codificación de marcas: en R m_o m_1

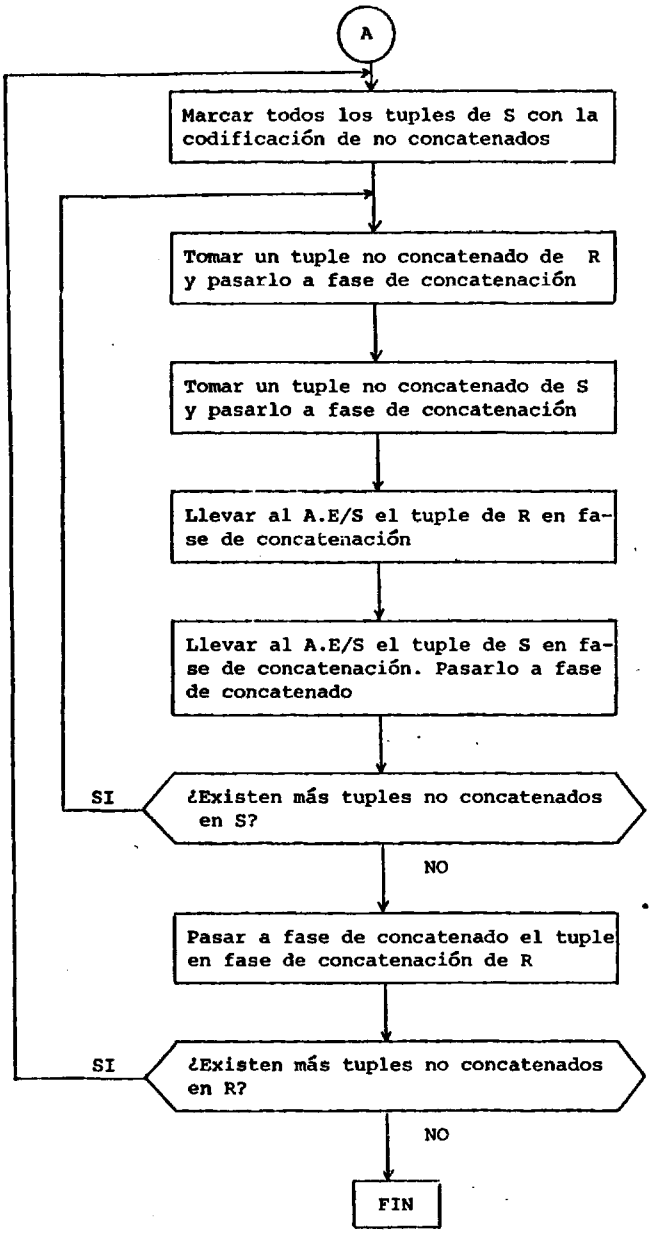
0	0	no concatenado con ningún tuple de S
0	1	en fase de concatenación con los tuples de S
1	0	concatenado con todos los tuples de S
1	1	no utilizada

en S m_o m_1

0	0	no concatenado con el tuple de R en fase de concatenación
0	1	en fase de concatenación con el tuple de R en fase de concatenación
1	0	concatenado con el tuple de R en fase de concatenación
1	1	no utilizada

Organigrama:




A
Marcar todos los tuples de S con la codificación de no concatenados
Tomar un tuple no concatenado de R y pasarlo a fase de concatenación
Tomar un tuple no concatenado de S y pasarlo a fase de concatenación
Llevar al A.E/S el tuple de R en fase de concatenación
Llevar al A.E/S el tuple de S en fase de concatenación. Pasarlo a fase de concatenado
SI
¿Existen más tuples no concatenados en S?
NO
Pasar a fase de concatenado el tuple en fase de concatenación de R
SI
¿Existen más tuples no concatenados en R?
NO
FIN

Programa PCBD:

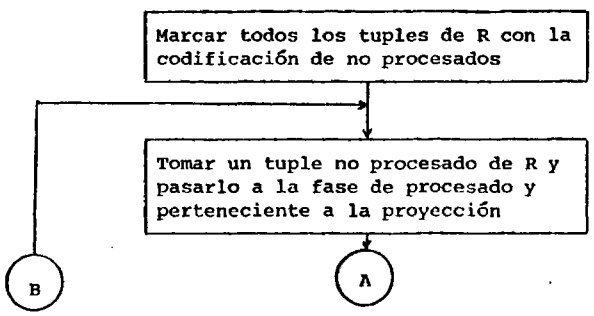
	CO	RELACION	DM	CUALIFICACION	P.E.
	[SD]	[R]	[$\overline{m_o}$ $\overline{m_1}$]	[Ø]	
1	[SD]	[S]	[$\overline{m_o}$ $\overline{m_1}$]	[Ø]	
	[RI]	[R($D_1,\ldots,D_n$)]	[$\overline{m_o}$ m_1]	[(M=00xx)]	[$r_1,\ldots,r_n$]
2	[RI]	[S($D'_1,\ldots,D'_m$)]	[$\overline{m_o}$ m_1]	[(M=00xx)]	[$r_1,\ldots,r_n$]
	[RE]	[R($D_1,\ldots,D_n$)]	[Ø]	[(M=01xx)]	[A.E/S]
	[RE]	[S($D_1,\ldots,D'_m$)]	[m_o $\overline{m_1}$]	[(M=01xx)]	[A.E/S]
	[BC]	[S]		[(M=00xx)]	[2]
	[SD]	[R]	[m_o $\overline{m_1}$]	[(M=01xx)]	
	[BC]	[R]		[(M=00xx)]	[1]
	[AL]				

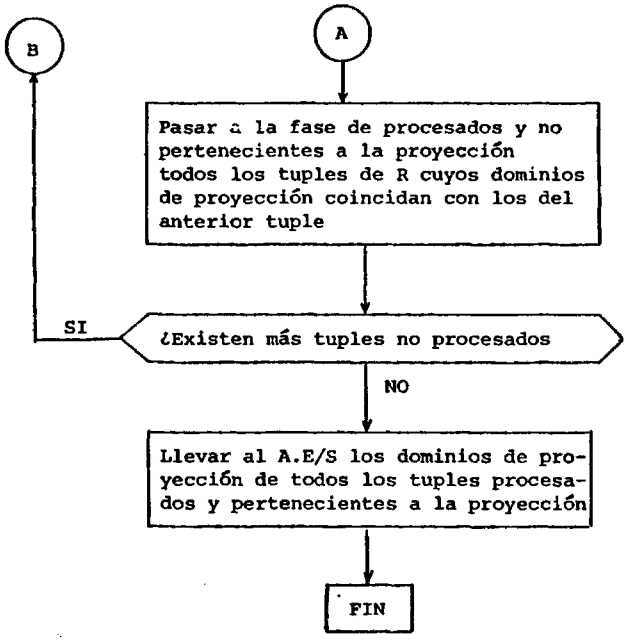
- PROYECCION: R[L]; R($D_1,\ldots,D_n$), L = {$D'_1,\ldots,D'_m$} {$D_1,\ldots,D_n$}, ($m \leq n$)

Codificación de marcas m_o m_1

0 0	tuples no procesados
0 1	tuples procesados y pertenecientes a la proyección
1 0	tuples procesados y no pertenecientes a la proyección
1 1	no se utiliza

Organigrama:





Programa PCBD:

	CO	RELACION	DM	CUALIFICACION	P.E.
	[SD]	[R]	$[\overline{m_1}\ \overline{m_2}]$	[ø]	
1	[RI]	$[R(D'_1,\ldots,D'_m)]$	$[\overline{m_o}\ m_1]$	[(M=00xx)]	$[r_1,\ldots,r_m]$
	[SD]	[R]	$[m_o\ \overline{m_1}]$	$[(M=00xx)\wedge(D'_1=<r_1>)\wedge\ldots\wedge(D'_m=<r_m>)]$	
	[BC]	[R]		[(M=00xx)]	[1]
	[RE]	$[R(D'_1,\ldots,D'_m)]$	[ø]	[(M=01xx)]	
	[AL]				

- COMPOSICION: $R[D_i \ \theta \ D'_j]S$; $R(D_1,\ldots,D_n)$, $S(D'_1,\ldots,D'_m)$

Al igual que en el Producto Cartesiano, en la Composición habrá que llevar al A.E/S la concatenación de tuples de R con tuples de S. Sin embargo, en esta operación relacional sólo se concatenan dos tuples de las

respectivas relaciones si cumplen la comparación θ sus dominios de composición $\mathcal{D}_i$ y $\mathcal{D}'_j$.

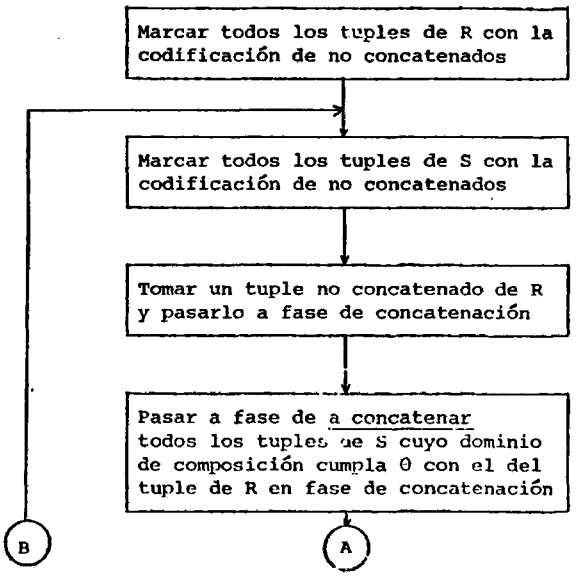
Calificación de marcas: en R m_o m_1

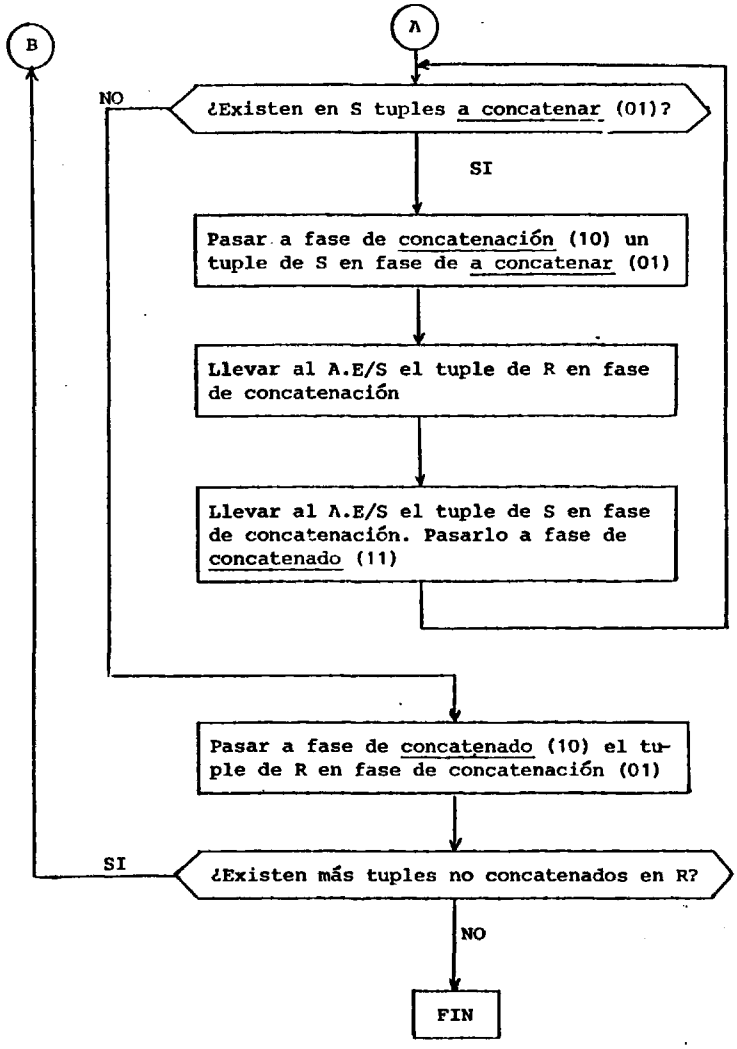
0	0	tuples no concatenados
0	1	tuple en fase de concatenación
1	0	tuples concatenados
1	1	no usada

en S m_o m_1

0	0	tuples no concatenados
0	1	tuples a concatenar con el tuple de R en fase de concatenación
1	0	tuple en fase de concatenación con el tuple de R en fase de concatenación
1	1	tuples concatenados con el tuple de R en fase de concatenación

Organigrama:




A
B
NO
¿Existen en S tuples a concatenar (01)?
SI
Pasar a fase de concatenación (10) un tuple de S en fase de a concatenar (01)
Llevar al A.E/S el tuple de R en fase de concatenación
Llevar al A.E/S el tuple de S en fase de concatenación. Pasarlo a fase de concatenado (11)
Pasar a fase de concatenado (10) el tuple de R en fase de concatenación (01)
SI
¿Existen más tuples no concatenados en R?
NO
FIN

	CO	RELACION	DM	CUALIFICACION	P.E.
	[SD]	[R]	$[\overline{m_o}\ \overline{m_1}]$	[ø]	
1	[SD]	[S]	$[\overline{m_o}\ \overline{m_1}]$	[ø]	
	[RI]	$[R(D_i)]$	$[\overline{m_o}\ m_1]$	[(M=00xx)]	$[r_i]$
	[SD]	[S]	$[\overline{m_o}\ m_1]$	$[(D'_j \theta < r_i >)]$	
2	[BC]	[S]		[(M=01xx)]	[3]
	[BC]	[ø]		[ø]	[4]
3	[RI]	$[S(D_k)]$	$[m_o\ \overline{m_1}]$	[(M=01xx)]	$[r_k]$
	[RE]	$[R(D_1,\ldots,D_m)]$	[ø]	[(M=01xx)]	[A.E/S]
	[RE]	$[S(D'_1,\ldots,D'_m)]$	$[m_o\ m_1]$	[(M=10xx)]	[A.E/S]
	[BC]	[ø]		[ø]	[2]
4	[SD]	[R]	$[m_o\ \overline{m_1}]$	[(M=01xx)]	
	[BC]	[R]		[(M=00xx)]	[1]
	[AL]				

- <u>RESTRICCION</u>: $R|D_i \theta D_j|$; $R(D_1,\ldots,D_n)$, $i \neq j$

Dada la posibilidad de las cualificaciones PCBD de expresar búsquedas por contexto de forma directa, la restricción relacional, tanto física como implícita, se reduce a una sola instrucción. Refiriéndonos al primer caso el programa PCBD sería:

[RE] $[R(D_1,\ldots,D_n)]$ [ø] $[|D_i \theta D_j|]$ [A.E/S]
[AL]

- <u>DIVISION</u>: $R[D_i \div D'_j]S$; $R(D_1,\ldots,D_n)$, $S(D'_1,\ldots,D'_m)$

Como recientemente demostró Healey ([24]), la división relacional es expresable en términos de las operaciones anteriores:

$$R[D_i \div D'_j]\ S = R[\overline{D_i}] - ((R[\overline{D_i}] \ \theta \ S[D'_j]) - R)\ [\overline{D_i}]$$

Por ello, su codificación con instrucciones PCBD no es necesaria para demostrar el carácter completo del repertorio.

II.5.- FUNCIONAMIENTO CONCURRENTE DEL PCBD

La organización MIMD del PCBD permite que cada una de sus células ejecute, en el mismo intervalo de tiempo, una cualquiera de las funciones elementales (primitivas) de su repertorio. Esta característica se explota a nivel de la U.C. para conseguir dos propiedades fundamentales del procesador:

1) acceso asociativo a nivel de relación: asignando la misma primitiva de búsqueda al conjunto de células que soportan dicha relación.
2) procesamiento concurrente de las instrucciones: asignando diferentes primitivas a los diferentes conjuntos de células que soportan cada una de las relaciones.

Todas las células funcionan síncronamente, es decir, sus respectivos elementos de memoria rotan sus datos bajo control de la misma base de tiempos. En un período de revolución, que denominaremos ciclo PCBD, cada célula ejecuta una primitiva de su repertorio recibida de la U.C. en el ciclo anterior. Merced al funcionamiento solapado de carga/ejecución, las células están disponibles para ejecutar primitivas en cualquier ciclo.

La ejecución de una instrucción PCBD comporta, en general, la ejecución de una o dos primitivas en las células que soportan la relación o relaciones referenciadas. Dependiendo de cada instrucción, estas pri-

mitivas habrán de ejecutarse durante un número determinado de ciclos. Atendiendo a ello podemos clasificar las instrucciones en tres grupos:

- Grupo 1.- Instrucciones que se ejecutan en la U.C. sin el concurso del conjunto de células (número de ciclos PCBD = 0).
- Grupo 2.- Instrucciones que se ejecutan a través de una primitiva en un único ciclo (número de ciclos PCBD = 1).
- Grupo 3.- Instrucciones que se ejecutan por medio de una o dos primitivas en un número de ciclos dependiente de los datos e indeterminado a priori (número de ciclos PCBD = f (datos)).

En la tabla de la figura II-9 se muestran los ciclos de ejecución para cada una de las instrucciones del repertorio.

Diremos que una célula está asignada a un programa PCBD cuando en ella se ejecuta una primitiva correspondiente a la instrucción en curso del citado programa. Para permitir una política interrumpible (preemptive) de asignación de células (recursos de proceso) a los N programas que concurrentemente se ejecutan en el PCBD con posibilidad de reasumir su ejecución sin pérdida del proceso realizado, las instrucciones del grupo 3 han de ser interrumpibles en cualquier ciclo, esto es, sus primitivas respectivas han de realizar procesos reasumibles a partir del último ciclo ejecutadas.

En el PCBD, esta característica se consigue haciendo que las primitivas procedentes de un mismo programa actúen sobre un único campo de marca de los N disponibles en cada relación. De esta forma, cada campo de marca mantendrá, en forma codificada, el resultado de los procesos selectivos que en cada instante llevan realizados los N programas en ejecución.

INSTRUCCIONES PCBD	N° de CICLOS	PRIMITIVAS DE CELULA
CREAR CR	1	LONGITUD DE TUPLE LT CODIGO DE RELACION CR
ELIMINAR EL	0	—
SELECCION DIRECTA SD	1	MARCA MA
RECUPERACION INTERNA RI	1	RECUPERACION RE
RECUPERACION EXTERNA RE	f(ntc)	SALIDA SA
BIFURCACION CONDICIONAL BC	1	MARCA MA
BORRADO BO	1	BORRADO BO
INSERCION IN	f(nti)	INSERCION IN
SUMA SU	1	SUMA SU
RESTA RS	1	RESTA RE
SUSTITUCION SS	1	SUSTITUCION SS
MAXIMO MA	1	MAXIMO MX
MINIMO MI	1	MINIMO MI
PROMEDIO PR	1	NUMERO-TOTAL NT
TOTAL TO	1	NUMERO-TOTAL NT
NUMERO NU	1	NUMERO-TOTAL NT
SELECCION INDIRECTA SI	f(ntc)	TRANSMISION TR RECEPCION RC
ESPACIO ES	1	ESPACIO ES
SALTO CONDICIONAL SC	0	—
ALMACENAMIENTO AM	0	—
ALTO AL	0	—
CARGAR CA	0	—

Fig. II-9.- Tabla de instrucción PCBD con indicación del número de ciclos de ejecución y primitivas de célula asociadas.

Las operaciones elementales que las primitivas realizan sobre los datos de los elementos de memoria de las células vienen impuestas por las exigencias semánticas de las correspondientes instrucciones PCBD y podemos resumir en:

- Determinación de los tuples que cumplen una cualificación: búsqueda por contexto.
- Modificaciones (suma, resta, etc.) de dominios
- Modificaciones de los bits de marca
- Funciones de agregación (máx, mín, etc.) sobre dominios
- Salida de datos seleccionados
- Intercambio de datos entre células
- Inserción de datos
- Control del espacio libre de memoria
- Adscripción de una célula a una relación determinada

Una primitiva será, pues, la definición del conjunto de tales operaciones que una instrucción PCBD necesita realizar, durante un ciclo, en una célula individual de las que componen la relación o relaciones referenciadas.

Antes de pasar a describir el repertorio de primitivas ejecutables en cada célula estudiaremos el comportamiento funcional de esta última.

II.5.1.- Comportamiento funcional de una célula

Cada célula se compone de una elemento de memoria y otro de procesamiento. El primero contiene tuples pertenecientes a una única relación y codificados según el formato descrito en el apartado II.3.1. El

segundo es un microprocesador no-numérico de propósito especial con capacidad para procesar todos los tuples del elemento de memoria durante un período de revolución.

Básicamente, cada microprocesador consta de una Lógica Operacional, donde tiene lugar el procesamiento de los tuples, y un Espacio de Memoria destinado a mantener la información que controla dicho procesamiento y a almacenar los resultados generados (Fig. II-10). La información de control define, tanto el significado del flujo serie de datos procedentes de la memoria rotante (formato) como las operaciones elementales que sobre ellos tienen lugar. La primera, representada por un conjunto de parámetros que denominaremos de reconfiguración, determina la pertenencia de la célula a una relación determinada. Los valores de estos parámetros se mantendrán, pues, invariables durante todo el tiempo que dura esta pertenencia. Los parámetros que definen las operaciones elementales, denominados parámetros operaciones, y los resultados de tales operaciones, se soportan sobre una zona del Espacio de Memoria compuesta por dos bloques idénticos (1 y 2 en la Fig. II-10) con funciones alternativas de carga y control. Esto es, mientras el bloque 1 dirige el procesamiento de la célula durante una revolución, el bloque 2 es accedido por la U.C. para recoger los resultados generados en la anterior revolución y cargar los valores de los parámetros operativos que definen las operaciones a ejecutar en la siguiente. Al finalizar toda revolución se conmutan las funciones de cada bloque.

Cada célula dispone, además, de dos vías de comunicación serie a través de las cuales se conectan a los buses de Salida e Intercambio. Por la primera transmitirá a la U.C. los datos seleccionados en las operaciones de salida. La segunda, bidireccional, permite la comunicación

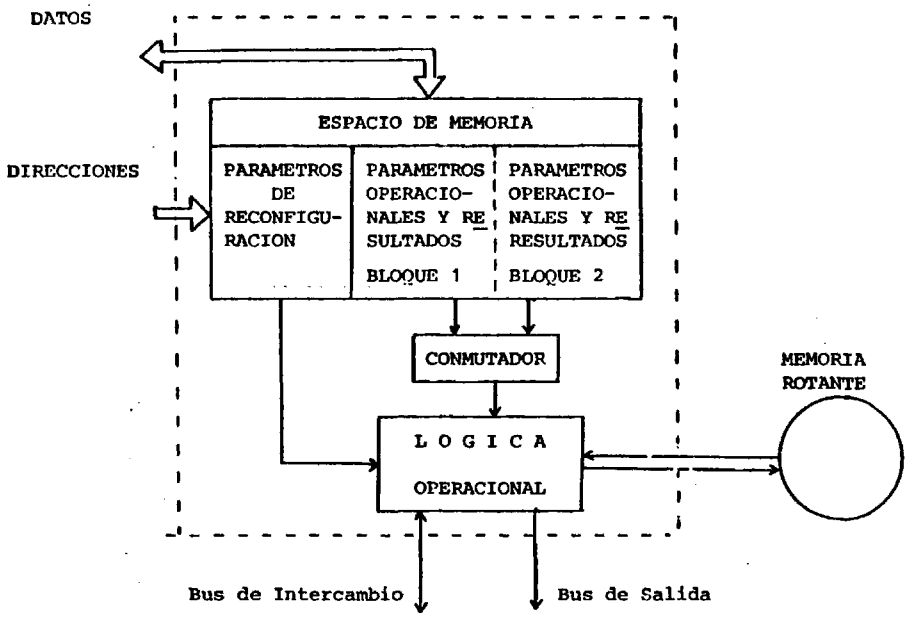


Figura II-10

entre dos conjuntos de células para el intercambio directo de datos exigido en la instrucción RI.

Las células son gobernadas desde la U.C. por medio de los buses de Direcciones y Datos. El primero permite la identificación de cada una de las células y la referenciación individualizada de las posiciones de su Espacio de Memoria. Por el bus de Datos se realizará el intercambio de información entre estas posiciones y la U.C. Las correspondientes a los parámetros operativos y de reconfiguración serán escritas en el curso de la carga de una primitiva; las que almacenan los resultados de operaciones, serán leídas una vez que la primitiva ha sido ejecutada. Estos últimos los denominaremos, en la descripción de las primitivas, registros de la célula.

II.5.1.1.- Primitivas de célula

Atendiendo al tipo de parámetros que modifican, distinguiremos dos grupos de primitivas:

a) Primitivas de Reconfiguración: asignan valores a los parámetros internos de la célula que determinan su adscripción a una relación determinada. Estos parámetros son dos: longitud de los tuples que soporta y una dirección codificada. El primero es utilizado por los circuitos de sincronismo de la célula en el seguimiento de los datos de su elemento de memoria. El segundo permite la referenciación de las células que soportan la misma relación. Las correspondientes primitivas son:

- LONGITUD (LT)

  formato: <LT> <VALOR>

  función: sincroniza el procesamiento de la información en el elemento rotante y formatea todas sus posiciones de tuple, es decir, desactiva los bits A. Estos serán los que se encuentren a distancias binarias múltiplo de VALOR.

- DIRECCION (DR)

  formato: <DR> <VALOR>

  función: posibilita la referenciación de la correspondiente célula por el VALOR binario asignado

Ambas primitivas se utilizan desde la U.C. en la implementación de la instrucción CREAR.

b) Primitivas Operativas: asignan valores a los parámetros de la célula que definen en ella una función de procesamiento: la correspondiente en dicha célula a la función global de una instrucción PCBD.

El contenido semántico coincidirá, pues, en buen número de ellas, con el de las instrucciones de procedencia. Su formato general es:

<CO> <CM> <DM> <C> <DO> <OP>

donde el Código de Operación (CO) especifica la función a realizar sobre los Dominios (DO) de los tuples de la célula que cumplan la Cualificación (C). En general, esta función implicará la presencia de un Operando (OP). Todos los tuples cualificados modificarán el Campo de Marcas (CM), según la Descripción de Marca (DM).

Los campos DM y C tendrán la misma forma que la descrita en las instrucciones PCBD. CM será un valor, en binario, del número de orden que el campo de marcas utilizado ocupa en la relación. Un dominio DO, vendrá definido por el número de orden de su primer byte y el número de éstos que lo componen. Finalmente, OP será una constante explícitamente declarada o bien un nuevo dominio, en las modificaciones por contexto. Las primitivas operativas son:

- MARCA (MA)

formato: <MA><CM><DM><C>

función: todos los tuples de la célula que cumplen C modifican CM según DM. Un bit del registro de estado (RE) de la célula (bit de cualificación) será activado si por lo menos existe un tuple que cumpla C. En caso contrario, dicho bit será desactivado.

utilización: a través de esta primitiva, la U.C. ejecuta las instrucciones de Selección Directa (SD) y Bifurcación Condicional (BC). Para esta última la "O" lógica de todos los bits de cualificación correspondientes a

las células de la relación referenciada, determinará la condición de salto global de la instrucción.

- RECUPERACION (RE)

formato: <RE> <CM> <DM> <C> <DO_s>

función: El primer tuple procesado de la célula que cumple C es marcado en CM según DM. Además, los dominios DO_s son recuperados en registros de la célula.

utilización: en la instrucción de Recuperación Interna (RI). Hay que hacer notar que al exigir esta instrucción la recuperación y marca de un único tuple en toda la relación, es necesario disponer de un mecanismo de comunicación entre las células que la soportan capaz de invalidar en éstas todo el proceso posterior al instante en que una cualquiera de ellas encuentre un tuple cualificado. En el apartado siguiente trataremos de este mecanismo.

- SALIDA (SA)

formato: <SA> <CM> <DM> <C> <DO_s>

función: los dominios DO_s de todos los tuples de la célula que cumplan C son transmitidos al bus de Salida. El campo CM de estos tuples es modificado según DM.

utilización: en la instrucción de Recuperación Externa (RE). Análogamente al caso anterior, es necesario disponer de mecanismos de comunicación intercelulares para evitar que más de una célula, con tuples cualificados ocupando la misma posición angular, accedan simultáneamente al bus de Salida.

- BORRADO (BO)

  formato: <BO> <C>

  función: el bit A de todos los tuples de la célula que cumpla C es desactivado. Estos tuples son inoperantes en procesos posteriores.

  utilización: en la instrucción de Borrado (BO)

- INSERCION (IN)

  formato: <IN> <OP>

  función: los tuples declarados en OP son almacenados en posiciones de tuple con el bit A desactivado; se activa el bit A de estas posiciones.

  utilización: en la instrucción de Inserción (IN). Para evitar que un mismo tuple se inserte repetidamente en la relación referenciada por la instrucción IN, una primitiva de inserción se ejecuta exclusivamente sobre una única célula.

- SUMA, RESTA, SUSTITUCION (SU, RE, SS)

  formato: $\left\{\begin{matrix}\text{<SU>}\\ \text{<RE>}\\ \text{<SS>}\end{matrix}\right\}$ <CM> <DM> <C> <DO> <OP>

  función: el dominio DO de los tuples de la célula que cumplen C, es sumado, restado o sustituido, respectivamente con (por) el operando OP. El campo CM de estos tuples es modificado según DM.

  utilización: en las instrucciones de Suma (SU), Resta (RS) y Sustitución (SS), respectivamente.

- MAXIMO, MINIMO, NUMERO-TOTAL (MX, MI, NT)

  formato: $\left\{\begin{matrix}\text{<MX>}\\ \text{<MI>}\\ \text{<NT>}\end{matrix}\right\}$ <CM> <DM> <C> <DO>

  función: con los dominios DO de todos los tuples de la célula que cumplan C se calcula su valor máximo, mínimo o número-total respectivamente. Además, los tuples cualificados son marcados en CM según DM. Los valores calculados se almacenan en registros internos de la célula.

  utilización: en las instrucciones Máximo (MA) y Mínimo (MI), respectivamente, las dos primeras primitivas; y en Promedio (PR), Total (TO) y Número (NU), la tercera.

- TRANSMISION (TR)

  formato: <TR> <CM> <DM> <C> <DO>

  función: el dominio DO de cada tuple de la célula que cumpla C es transmitido al bus de Intercambio. Dichos tuples son modificados en CM según DM.

- RECEPCION (RC)

  formato: <RC> <CM> <DM> <OP>

  función: dispone a la célula para aceptar valores de dominio DO_i procedentes del bus de Intercambio y formar en ella cualificaciones disyuntivas de la forma:

$$(DR\ \theta\ DO_1)\ V\ (DR\ \theta\ DO_2)\ V\ \ldots\ V\ (DR\ \theta\ DO_p)$$

En la revolución siguiente marcará CM según DM de todos los tuples de su elemento de memoria asociado que cumplan dicha cualificación. El operando OP definirá el dominio DR que interviene en la cualificación, así como el número máximo p de valores DO_i que es capaz de aceptar en cada revolución. Este número, como veremos en el capítulo III, dependerá de la longitud de los tuples que soporta.

utilización de las primitivas TR y RC: la U.C. las utiliza conjuntamente en la realización de la instrucción de Selección Indirecta:

[SI] [R1] [DM1] ((D1 Θ D2) [R2] [DM2] [C])

para ello, asigna a las células que soportan las relaciones R1 y R2, respectivamente, las primitivas RC y TR con los siguientes valores para los parámetros:

R1 ⟵ <RC> <CM> <DM1> <D1p>

R2 ⟵ <TR> <CM> <DM2> <C> <D2>

en el ciclo PCBD siguiente lás células de R2 cualificarán sus tuples con C y transmitirán valores de D2 a las células de R1 a través del bus de Intercambio. Simultáneamente, las células de R1 construyen con estos valores una cualificación disyuntiva que aplicarán, en el ciclo siguiente, a sus tuples.

En el diagrama de la figura II-11 se muestra la uti-

lización de las células respectivas de R1 y R2 en el curso de la ejecución de RI. En cada ciclo CI se especifica la función de carga (C) y ejecución (E), así como la disponibilidad de las células por la U.C., una vez comenzada la ejecución.

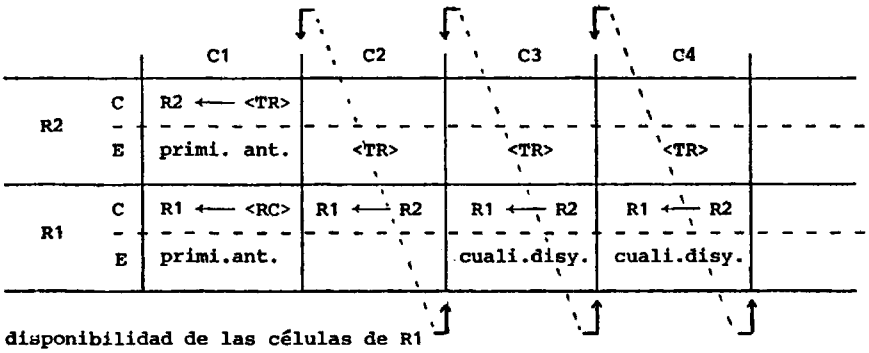


Figura II-11

- ESPACIO (ES)

  formato: <ES>

  función: determina el número de posiciones de tuple libres (bit A = 0) de la célula y los almacena en un registro.

  utilización: en la instrucción Espacio (ES).

II.5.1.2.- Dependencias entre los procesos celulares

Desde un punto de vista operativo, los procesos que se lleva a efecto en las diferentes células que soportan una relación, en el curso

so de la realización de una instrucción, pueden ser independientes o dependientes entre sí. Los primeros se resuelven completamente en cada una de las células, sin condicionamiento de lo que ocurre en las restantes. Así, los procesos de marca de la instrucción SD son independientes. También lo son los relativos a las instrucciones de agregación. Cada célula evalúa la función correspondiente sobre el subconjunto de datos que soporta, siendo la U.C. quien elabora la función global sobre la relación a partir de los datos parciales obtenidos de las diferentes células.

Los procesos dependientes, ya reseñados en la utilización de las primitivas afectadas, son motivados por dos tipos de causas: la demanda simultánea de un recurso único por parte de dos o más células (primitivas de salida y transmisión) o bien, la propia naturaleza de la instrucción PCBD al exigir la actuación sobre un único tuple de la relación cuya localización celular es desconocida (primitiva de recuperación).

Estos procesos tienen, no obstante, un denominados común: su realización está condicionada al cumplimiento de una cualificación. Ello significa que con posterioridad a la constatación de tal cumplimiento y antes de desencadenar el proceso, cada célula debe asegurarse de ser la única que lo realiza. La solución de este problema pasa necesariamente por el establecimiento de un control centralizado que arbitre los casos de simultaneidad.

El método que hemos adoptado para el PCBD es el conocido como selección por encadenamiento de prioridad (daisy-chaining) ([25]). Básicamente consta de tres líneas de control conectadas a todas las células del CC y a una Unidad denominada de Gobierno, que consideraremos localizada en la U.C., aunque su función es independiente (Fig. II-12). Cuando

en una célula un tuple cumpla la cualificación generará una señal de Petición a la Unidad de Gobierno en demanda de autorización para realizar

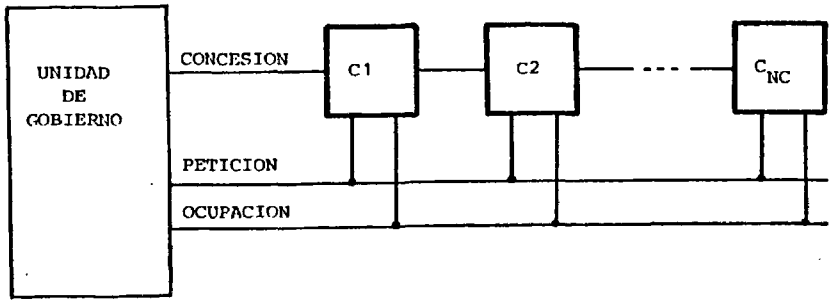


Figura II-12.- Selección por encadenamiento de prioridad (daisy-chaining)

el proceso dependiente. Simultáneamente bloqueará la línea de Concesión para todas las células de la cadena posteriores a ella. La Unidad de Gobierno, si la línea de Ocupación está desactivada, responderá a ésta y todas las peticiones que se hallan podido efectuar simultáneamente enviando una señal por la línea de Concesión. Esta, tan sólo llegará a la célula, de entre todas las que hicieron la petición, más próxima a la Unidad de Gobierno. Dicha célula activará la línea de Ocupación a fin de evitar la concesión de nuevas peticiones mientras dura la realización del proceso.

Cada uno de los tres procesos dependientes podrán tener lugar simultáneamente sobre tres conjuntos de células diferentes. Por ello, cada célula (Fig. II-13) será recorrida por tres conjuntos independientes de líneas de Concesión (C), Petición (P) y Ocupación (O).

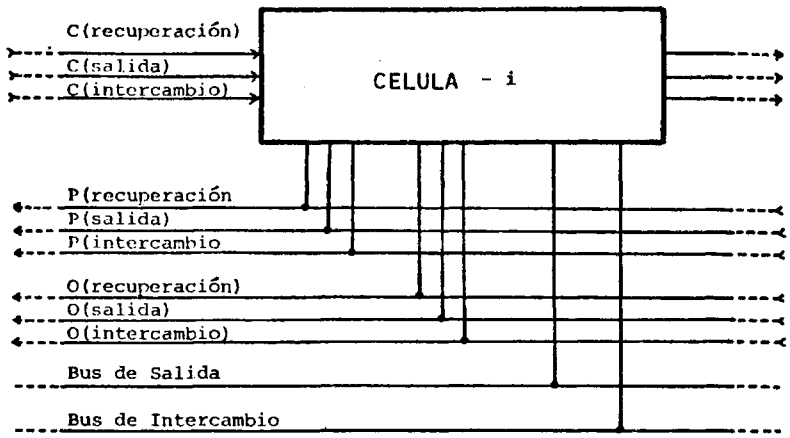


Figura II-13

II.5.2.- Gestión de los recursos de proceso: Unidad de Coordinación

Los recursos de proceso gestionados para la U.C. en la ejecución de los N programas PCBD que en ella residen los componen las NC células, el bus de Intercambio (I) y el bus de Salida (S). Cuando un recurso es utilizado por una primitiva que corresponde a la instrucción en curso de un programa, diremos que está asignado a dicho programa. Evidentemen, los recursos I y S habrán de asignarse asociados a un conjunto de células.

En virtud de la descomponibilidad en ciclos de todas las instrucciones PCBD, a excepción de la RI que exige, como vimos en el apartado anterior, dos ciclos consecutivos de las células receptoras, la asignación de los recursos de proceso a los N programas en ejecución se hará dinámicamente, a intervalos de tiempo de un ciclo, siguiendo una políti-

ca que obtenga la mejor capacidad de servicio del sistema.

Cada aplicación particular, en función de las exigencias de su contexto de funcionamiento, caracterizará, de forma diferente, la capacidad de servicio de su sistema. Así, habrá aplicaciones que requieran una capacidad de procesamiento (throughtput) máxima independientemente de los retardos que ello pueda ocasionar a determinados accesos a la base. En este caso, la política de asignación se hará de forma que en cada ciclo se utilice el máximo número de recursos de proceso. En otras aplicaciones interesa, por el contrario, proporcionar tiempos iniciales de respuesta pequeños para la totalidad de los usuarios de la base, es decir, minimizar las diferencias de tiempo existentes entre los comienzos de respuesta de sus accesos respectivos. La política de asignación de recursos, adecuada a estos objetivos, deberá tender a no ejecutar instrucciones de RE de ningún programa PCBD hasta tanto no se encuentren todos en fase de ejecución de las mismas y, a partir de este momento, hacer un uso compartido del bus de Salida. Puede practicarse una política de asignación adecuada a una disciplina FIFO (First In, First Out) de servicio de accesos, esto es, establecer un orden de prioridad fijo en los programas PCBD, coincidente con el orden de llegada. Es decir, que para cada caso la política adecuada será diferente.

A fin de permitir el establecimiento de la política de asignación de recursos que en cada aplicación particular sirva del mejor modo posible a sus necesidades específicas, la U.C. será soportada sobre un procesador de propósito general. De esta forma, la política de asignación será implementada por software, con la inherente flexibilidad que ello le confiere.

Aparte de la asignación de recursos, la U.C., como vimos en el apartado II.3.2, tiene otros cometidos. De forma esquemática, sus responsabilidades podemos resumirlas en cuatro:

1) Recoger los accesos compilados en forma de programas PCBD de la memoria central del ordenador principal.
2) Ejecutarlos según la política de asignación de recursos establecida.
3) Recuperar los datos seleccionados en cada uno de ellos.
4) Transmitirlos a las áreas de E/S respectivas en memoria central del ordenador principal.

Las dos primeras funciones se realizan en el Controlador de Recursos y las dos restantes en el Controlador de Salida, que serán los dos componentes en que consideraremos dividida la U.C. (Fig. II-14) El primero lo constituye el procesador de propósito general, anteriormente mencionado, junto con su memoria central. El segundo será un buffer y su circuitería de control con funcionamiento autónomo en la recepción de datos de las células, pero inicializado por el controlador de Recursos.

II.5.2.1.- Controlador de Salida

Su función consiste en aceptar valores de dominio en forma serie a través del bus de Salida, almacenarlos temporalmente en un "buffer" y transmitirlos a la memoria central del ordenador principal vía ADM (acceso directo a memoria).

Cuando el PCBD ejecuta una instrucción de RE (Recuperación Externa) sobre una relación, el Controlador de Recursos inicializa:

a) El conjunto de células que soportan dicha relación con la

primitiva de Salida (SA).

b) El Controlador de Salida con la definición del Area de E/S especificado en RE, esto es, la dirección inicial y número de palabras referidos a memoria central del ordenador principal.

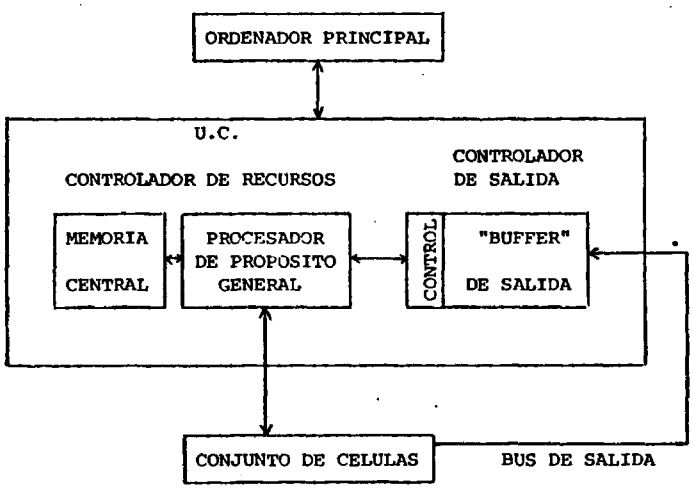


Fig. II-14.- Estructura general interna de la Unidad de Coordinación

En los sucesivos ciclos de la memoria rotante, las células transmiten valores de dominio de tuples que cumplen la cualificación de RE, al "buffer" de salida. Este proceso se interrumpe cuando:

a) La instrucción de RE finaliza, habiéndose transmitido todos los tuples cualificados. El Controlador de Salida transfiere el contenido del "buffer" al A.E/S.

b) El "buffer" de salida se llena. El Controlador de Recursos

interrumpe la salida de datos de las células hasta hacerse la transferencia del "buffer" al ordenador principal.

c) El Controlador de Recursos decide asignar el bus de Salida a otro programa PCBD. El contenido del "buffer" se transmite al ordenador principal y se guarda en el Controlador de Recursos la definición del A.E/S pendiente, hasta que de nuevo se reanudę la instrucción RE.

II.5.2.2.- Controlador de Recursos

En cada ciclo, el Controlador de Recursos establece una correspondencia entre el conjunto de instrucciones actuales de cada programa PCBD y los recursos de proceso disponibles: células, bus I y bus S.

Tres tipos de información contendrá la memoria central del Controlador de Recursos (Fig. II-15).

MEMORIA CENTRAL DEL CONTROLADOR DE RECURSOS

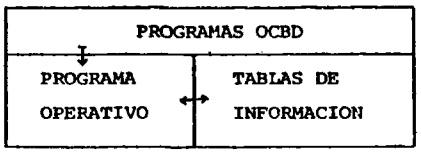


Fig. II-15.- Mapa de memoria del Controlador de Recursos

a) Conjunto de programas codificados con instrucciones PCBD recibidos del ordenador principal.

b) Conjunto de Tablas de Información referentes a la distribución celular de relaciones, estado de ejecución de los programas, órdenes de prioridad, etc., necesarias para la asignación de recursos.

c) Programa Operativo que implementa la política de asignación de recursos establecida

Las informaciones a) y b) serán, pues, los datos para el Programa Operativo. Esto es, los programas PCBD son interpretados con ayuda de las Tablas de Información por el Programa Operativo, único que físicamente se ejecuta en el procesador general y que utiliza el conjunto de células como unidades de E/S.

Un organigrama general de las tareas llevadas a cabo por el Programa Operativo en un ciclo PCBD se muestra en la Fig. II-16. Tras el análisis de las N instrucciones actuales de los programas PCBD, se determina, para cada una de ellas, los recursos de proceso, células y buses, que necesitan. Cuando más de una instrucción requiere para su realización un mismo recurso, éste se le asignará a la correspondiente al programa PCBD que resulte más prioritario al aplicar la política de asignación establecida. Una vez creada la correspondencia entre programas PCBD y recursos, el Programa Operativo construirá el conjunto de primitivas y lo distribuirá por las células. Ejecutadas las primitivas, actualizará las Tablas de Información con los resultados obtenidos. Si algún programa PCBD finaliza, interrumpirá al ordenador principal indicando su disponibilidad para aceptar uno nuevo.

Una política de asignación de recursos que permite poner de manifiesto, de forma clara, las ventajas de la organización MIMD del PCBD frente a la SISD de otrosprocesadores, resulta de aplicar una disciplina tipo FIFO al flujo de accesos que llegan al PCBD, con restricciones de precedencia para los accesos de actualización. Esta política podemos definirla en los siguientes términos:

a) A todos los programas recibidos del ordenador principal se le asigna un orden de prioridad coincidente con el orden de llegada.

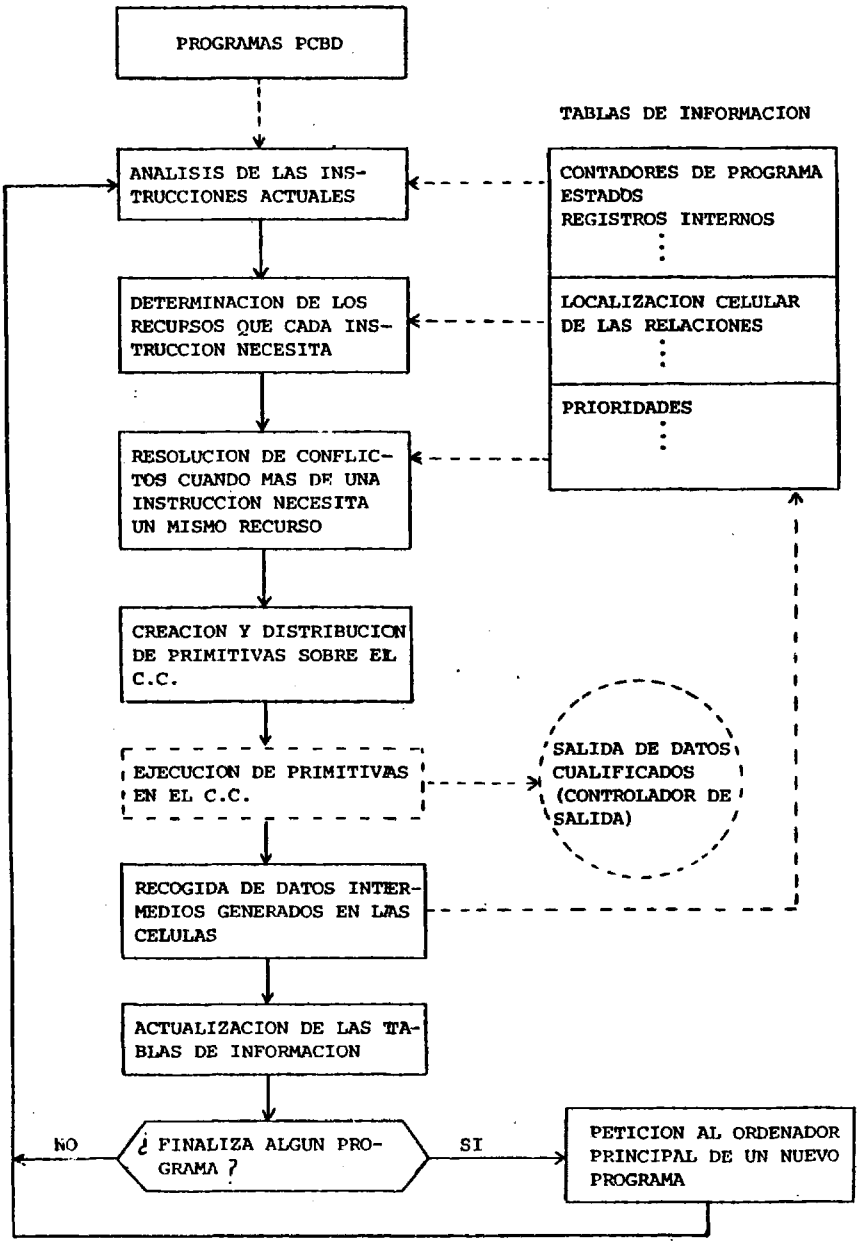


Fig. II-16.- Organigrama General del Programa Operativo

b) Los recursos de proceso demandados en cada ciclo se asignan siguiendo el orden de prioridad.

c) Un programa correspondiente a un acceso de actualización de la base (inserción, modificación o borrado) bloquea todas las relaciones en él referenciadas, para todos los accesos de menor prioridad, y le son bloqueados los referenciados por accesos de mayor prioridad.

En su implementación, el Programa Operativo mantendrá, para cada programa PCBD en ejecución, los recursos bloqueados por los accesos de actualización (Tabla de Recursos Bloqueados, Fig. II-17). También mantendrá, en la Tabla de Recursos Asignados, la contabilidad de los disponibles en cada instante de un ciclo PCBD.

Un organigrama de la fase de asignación de recursos correspondiente a esta política se muestra en la figura II-17.

Con esta política, el programa, más prioritario P1, dispondrá en cada ciclo PCBD de la totalidad de los recursos de proceso, es decir, se ejecutará a la misma velocidad que lo hiciera en una organización SISD. Además, los recursos no utilizados se irán asignando a los sucesivos programas en orden ascendente de prioridad. De esta forma, cuando el programa P1 finaliza, los restantes programas P2,...,PN se encontrarán en una fase de ejecución más o menos avanzada. La cantidad de procesamiento realizada sobre P2,...,PN al finalizar P1 será una medida de las ventajas de la organización MIND frente a la SISD. La Figura II-18 muestra, en forma gráfica, la ejecución concurrente de tres programas P1, P2 y P3, siguiendo la citada política de asignación. Para cada ciclo CI se indica la asignación de recursos a los programas, esto es,

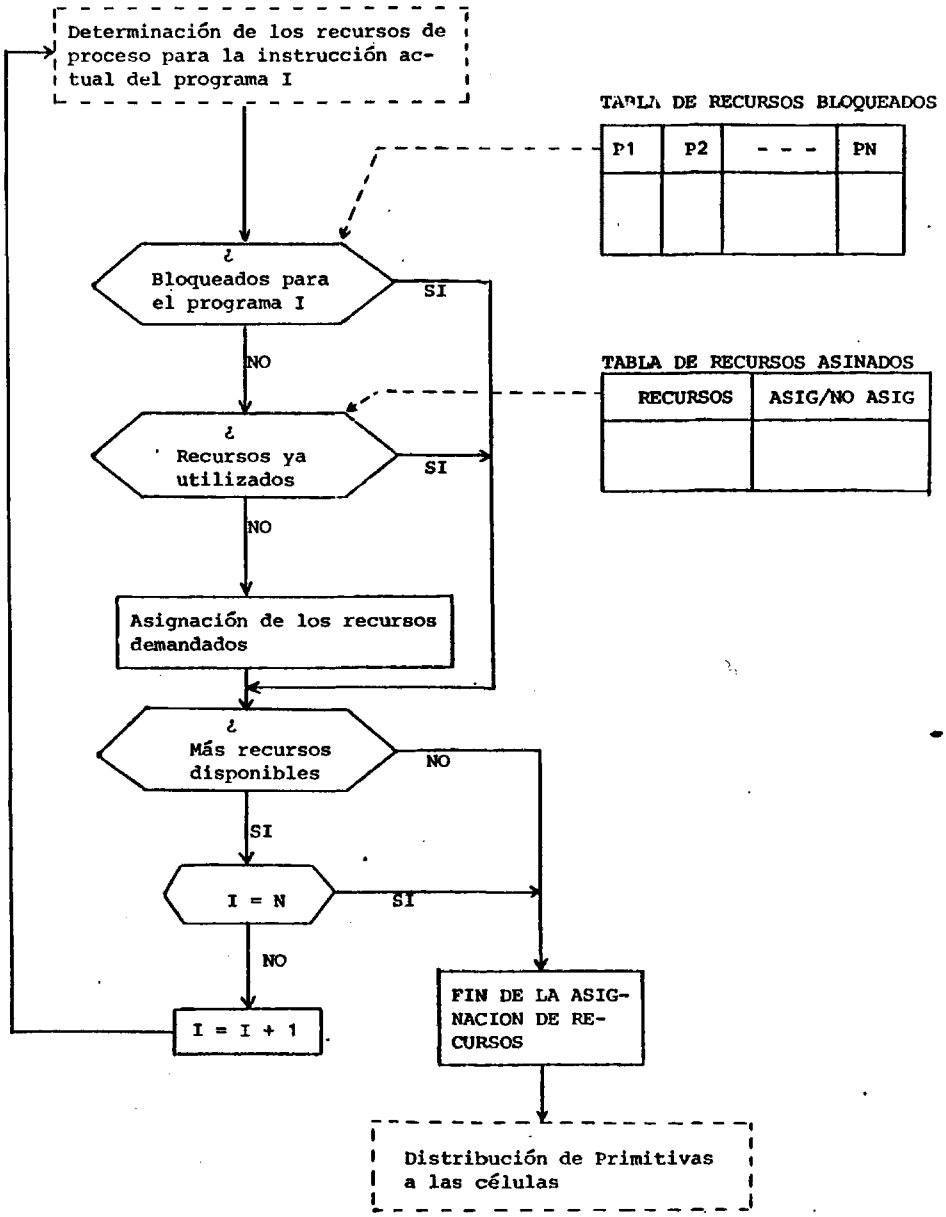

Determinación de los recursos de proceso para la instrucción actual del programa I
TABLA DE RECURSOS BLOQUEADOS
P1
P2
- - -
PN
¿ Bloqueados para el programa I
SI
NO
TABLA DE RECURSOS ASINADOS
RECURSOS
ASIG/NO ASIG
¿ Recursos ya utilizados
SI
NO
Asignación de los recursos demandados
¿ Más recursos disponibles
NO
SI
I = N
SI
NO
I = I + 1
FIN DE LA ASIGNACION DE RECURSOS
Distribución de Primitivas a las células

Fig. II-17

células de las relaciones RI(I = 1,...,4), bus S y bus I. A la derecha de cada instrucción hemos especificado el número de clicos PCBD que dura su ejecución.

PROGRAMA P1	N.C.
[SD] [R1]	1
[SI] [R1,R2]	3
[RE] [R2]	3
[AL]	0

PROGRAMA P2	N.C.
[MA] [R3]	1
[SD] [R1]	1
[SD] [R1]	1
[SI] [R1,R4]	2
[RE] [R4]	1
[AL]	0

PROGRAMA P3	N.C.
[PR] [R2]	1
[SD] [R3]	1
[SD] [R3]	1
[RE] [R4]	4
[AL]	0

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
P3	R2	R3	R3	R4,S	R4,S					R4,S		R4,S
P2	R3					R1	R1	R1, R4,I	R1 R4,I	R1,I	R4,S	
P1	R1	R1 R2,I	R1 R2,I	R1 R2,I	R1	R2,S	R2,S					

FIN DE P1 FIN DE P2 FIN DE P3

Fig. II-18

Es evidente que, cuanto mayor sea la diversificación de las referencias en los programas PI, mayor será el rendimiento conseguido por el PCBD. Una adecuada política en este sentido, mantenida en el ordenador principal respecto a los accesos enviados al PCBD, mejorará la performance global del sistema.

REFERENCIAS

(1) PARKER, J.L., "A logic-per-track retrieval system". Proc. of the IFIPS Conference pp TA-4-146 to TA-4-150, 1971.

(2) BAUM, R.I., HSIAO, D.K., and KANNAN, K., "The architecture of a database Computer - Part I: Concepts and Capabilities", The Ohio State Univ., Columbus, Tech. Rep. OSU-CISRC-TR-76-1, Sept. 1976.

(3) MINSKY, N., "Rotating storage devices as partially associative memories". Proc. of 1972 FICC, pp. 587-596.

(4) LUQUE, E., RUZ, J.J. y HERMIDA, R., "Dispositivos para operaciones entre conjuntos orientados a sistemas de memoria inteligentes". Actas del 4° Congreso de Informática y Automática. Madrid, Octubre 1979.

(5) LEILICH, H.O., STIEGE, G. and ZEIDLER, H. Ch., "A search processor for data base management systems". in 4th Int. Conf. on Very Data Bases, ACM, Berlin, Sept. 1978. pp. 280-287.

(6) CANADAY, R.H., "A back-end computer for data base management". CACM Vol. 17, n° 10, Oct. 1974, pp. 575-582.

(7) SLOTNICK, D.L., "Logic per track devices". Advances in Computers, vol. 10, New York, Academic Press 1970, pp. 291-296.

(8) OZKARAHAN, E.A., SCHUSTER, S.A., ans SMITH, K.C., "RAP-An Associative processor for data base management". Proc. of the 1975 NCC, pp. 379-386.

(9) LIN, C.S., SMITH, D.C., and SMITH, J.N., "The design of a rotating associative memory for relational database applications". ACM TODS, Vol. 1, pp. 53-65, Mar. 1976.

(10) SU, S.Y.W. and LIPOVSKI, G.T., "CASSM: A cellular system for very large data bases" in Proc. 1st. Int. Conf. on very Large Data Bases, ACM, N.Y., Sept. 1975, pp. 456-472.

(11) PARHAMI, B., "A highly parallel computing system for information retrieval". Proc. of 1972 Fall Joint Computer Conf., pp. 587-596.

(12) HSIAO, D.K. and MADNICK, S.E., "Database Machine architecture in the context of information technology evaluation", in Proc. 3rd Int. Conf. on Very Large Data Bases, ACM, N.Y., 1977, pp. 63-84.

(13) DEWITT, D.J., "DIRECT: A multiprocessor organization for supporting relational data base manogement systems". 5th Annual Symposium on Computer Architecture, 1978, pp. 182-189.

(14) YPMA, J.E., "Bubble domain memory systems". AFIPS Conf. Proc., vol. 44, pp. 523-528, May 1975.

(15) AMELLO, G.F., "Charge-coupled devices for memory applications". AFIPS Conf. Proc., vol. 44, pp. 515-522, May 1975.

(16) ZAKY, S.G., "Microprocessors for Non-Numeric Proccessing". Proc. of Third Work Shop on Computer Architecture for Non-Numerica Processing, May 1977, pp. 23-20.

(17) COPELALAND, G.P., LIPOVSKI, G.J., and SU, S.Y.W., "The architecture of CASSM: A Cellular System for Non-Numeric Processing". Proc. of the First Annual Workshop on Computer Architecture, 1973.

(18) PARHAMI, B., "RAPID; a rotating associative processor for information dissemination". Tech. Report UCLA-ENG-7213 University of California. Febr. 1972.

(19) LUQUE, E., RUZ, J.J., and BAUTISTA, A., "Database Concurrent Processor", in 5th Int. Conf. on Very Large Data Bases, ACM, Río de Janeiro, Oct. 1979.

(20) LUQUE, E., RUZ, J.J., and RIPOLL, A., "An equipment specially designed for the relational management of data". Proc. Convention Informátique, París, Sept. 1979, pp. 149-157.

(21) SCHUSTER, S.A., NGUYEN, H.B., OZKARAHAN, E.S., and SMITH, K.C., "RAP.2 - An associative Processor for Data Bases". in 5th Annual Symposium on Computer Architecture; ACM, April 1978, pp. 52-59.

(22) OZKARAHAN, E.A., DOGAC, A., "Database processors used in Information Systems", Proc. Convention Informatique on Database Design, Paris, Sept. 1978.

(23) OZKARAHAN, E.A., and OFLAZER, K., "Microprocessor Based modular database processor", in 4th Int. Conf. on Very Large Data Bases, ACM, Berlin, Sept. 1978, pp. 300-311.

(24) HEALEY, P., IBM Research, San José RJ987.

(25) HAYES, J.P., "Computer Architecture and Organization". McGraw-Hill, 1978, pp. 413-414.

CAPITULO III

IMPLEMENTACION FISICA DEL PROCESADOR

III.1.- INTRODUCCION

Las características funcionales del PCBD, estudiadas en el Capítulo anterior, se han establecido teniendo presente, en todo momento, las actuales posibilidades de la tecnología hardware. Ello significa que muchos de los factores de implementación, que trataremos en el presente capítulo, han influenciado su estructura conceptual.

En la implementación física, la mayor complejidad gravita sobre los microprocesadores no-numéricos del Conjunto de Células. Sin embargo, hay que subrayar el carácter repetitivo de estos elementos con la favorable repercusión que ello supone a la hora de una realización. Además, el número de elementos activos y líneas de comunicación externa de los mismos se mantiene dentro de los límites que exigen las actuales técnicas para ser posible su integración en un único "chip". No obstante, su construcción a partir de componentes MSI y LSI, actualmente en el mercado, resulta perfectamente factible ([1]).

Las funciones de la Unidad de Coordinación, como hemos mencionado en el capítulo anterior, se soportan sobre un procesador de propósito general al que se conectan las células como periféricos convencionales. Para ello se puede utilizar un microprocesador "bit slice" completamente compatible con algún miniordenador, que disponga de un software potente (por ejemplo PDP-11), a fin de facilitar la codificación de los programas correspondientes, especialmente el que implementa la política de asignación de recursos.

III.2.- ESTRUCTURA INTERNA DE LAS CELULAS

La estructura interna de las células del PCBD deriva de las funciones que realiza y de la naturaleza rotante de los elementos de memoria. En general, el proceso global P que una célula lleva a efecto en la ejecución de una primitiva sobre cada tuple T de su elemento de memoria asociado, podemos descomponerlo en la suma de los siguientes subprocesos sobre T:

$$P(T) = C(T) + A(T) + S(T) + I(T) + E(T)$$

donde: $C(T)$ determina el cumplimiento o no de una Cualificación por T

$A(T)$ realiza las operaciones de tipo Aritmético, sobre dominios de T exigidas por las primitivas de modificación y agregación.

$S(T)$ transmite dominios de T hacia el bus de Salida

$I(T)$ transmite dominios de T hacia el bus de Intercambio

$E(T)$ sustituye campos de marca y dominios de T antes de proceder a su Escritura en el elemento de memoria. También inserta tuples nuevos.

Todos estos subprocesos han de realizarse a la velocidad de paso de T por la célula y no son, en general, independientes:

- el resultado de C condiciona la ejecución de los subprocesos S, T y E
- los datos generados en A deben estar disponibles antes de comenzar E
- A no está condicionado por C, puesto que la toma en consideración de los dominios operados es responsabilidad de E

Las dependencias anteriores obligan a la descomposición tem-

poral del proceso P en dos fases consecutivas:

$$F_1(T) = C(T) + A(T)$$

$$F_2(T) = S(T) + I(T) + E(T)$$

Los subprocesos componentes de cada una de estas fases, al ser independientes entre sí, podrán realizarse simultáneamente, es decir, en paralelo. No obstante, la fase F_2 no podrá comenzar hasta que la F_1 haya finalizado completamente, será pues necesario retardar la llegada de T a F_2 hasta que haya sido procesado completamente en F_1. En la figura III-1, se representa el diagrama de flujo de datos y procesamiento de una célula.

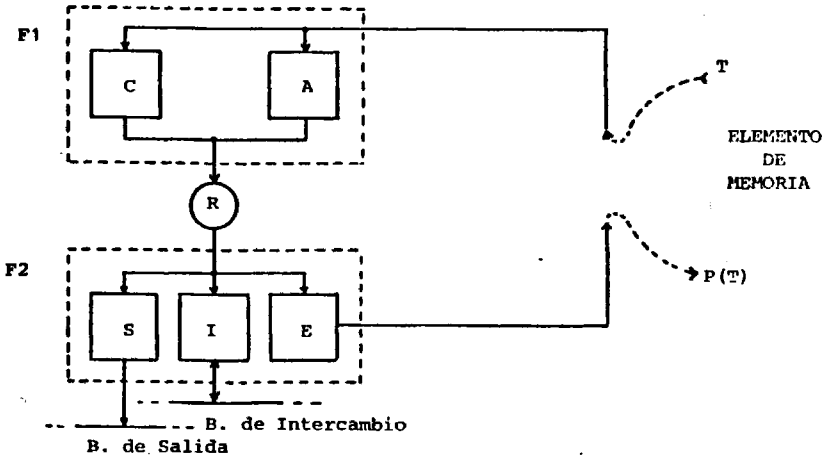


Figura III-1.- Flujo de datos y procesamiento de una célula

Una célula se organizará, pues, internamente como una estructura del tipo "pipe-line" con dos fases de procesamiento, componiéndose cada fase de procesos paralelos.

III.2.1.- Elementos de memoria

Los tuples de cada célula están soportados sobre un registro de desplazamiento rotante de 1/2 M bits de capacidad, construido con tecnología CCD ([2]). El acceso (lectura y escritura) a su información serie tiene lugar a través de una única posición de bit (Fig. III-2).

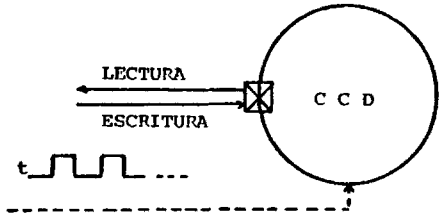


Figura III-2.- Elemento de memoria de una célula

La información rota bajo control de una base de tiempo t a una frecuencia que dependerá de las características de la memoria utilizada, pero comprendida entre 1 MHz. y 10 MHz.

También se podría utilizar para tal función memorias de tecnología "bubble" ([3]). Desde un punto de vista lógico su comportamiento es idéntico al que presentan los CCD; también su capacidad de almacenamiento es comparativa, no obstante, al ser su frecuencia de funcionamiento un orden de magnitud inferior, la velocidad de procesamiento de la célula no se aprovecharía al máximo.

III.2.2.- Elementos de procesamiento

El diagrama general de bloques de cada microprocesador no-numérico ([4]) o elemento de procesamiento de las células se muestra en la figura III-3.

Se compone de cinco unidades básicas: Cualificación, Aritmé-

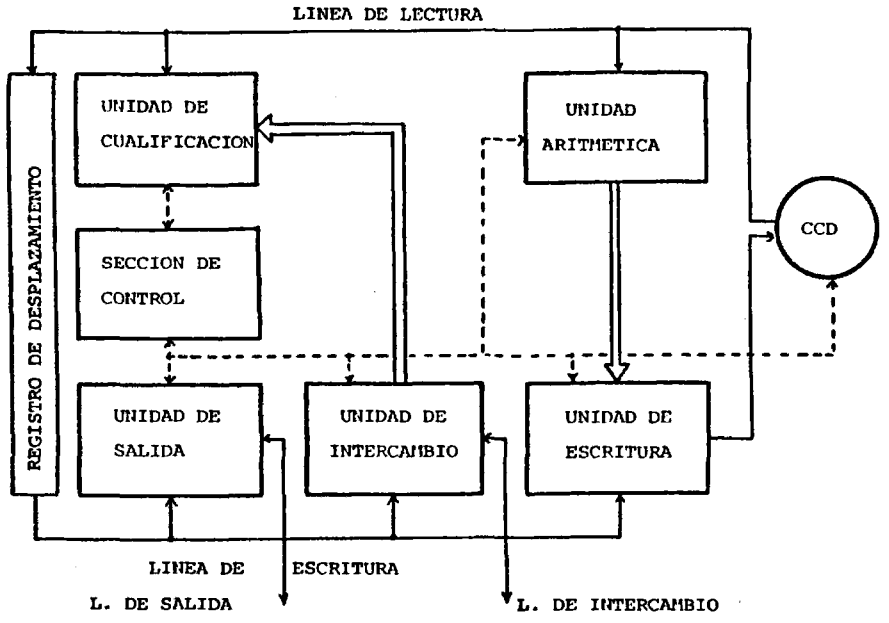


Figura III.3.- Diagrama de Bloques de una célula

tica, Salida, Intercambio y Escritura, coordinadas por la sección de Control. Las dos primeras actúan en paralelo en la fase F_1 de procesamiento de un tuple que denominaremos fase de lectura. Las tres restantes lo harán en la fase F_2 o fase de escritura. Un Registro de Desplazamiento serie con capacidad para contener un tuple de máxima longitud permitida en el PCBD, retardará el comienzo de la fase de escritura hasta que la de lectura haya finalizado.

Las diferentes posiciones del Espacio de Memoria correspondientes a los parámetros de funcionamiento y resultados generados, las consideraremos localizadas en la unidad sobre la que actúan.

Supuesto inicializado con todos los valores de los parámetros

correspondientes a una primitiva, el comportamiento funcional del microprocesador podemos describirlo como sigue. A la velocidad de desplazamiento de la CCD, la Unidad de Cualificación compara secuencialmente, en el orden de llegada, los dominios de un tuple con las constantes de Criterio de búsqueda. Paralelamente, si se trata de alguna primitiva de agregación o modificación, la Unidad Aritmética operará sobre el correspondiente dominio. Para el último caso, el dominio modificado es transmitido a la Unidad de Escritura. El Registro de Desplazamiento desfasa la llegada del tuple a la línea de escritura, esto es, cuando se conoce el resultado de la evaluación de la cualificación todo el tuple se halla contenido en él. En este punto, dependiendo de la primitiva en ejecución, determinados dominios del tuple podrán derivarse hacia la línea de salida o de intercambio por las correspondientes unidades; paralelamente, la Unidad de Escritura, previa sustitución de campos de marca y dominios, si la cualificación ha sido satisfecha, reescribirá de nuevo el tuple en la CCD.

En la Sección de Control se generan todas las señales (microórdenes) que gobiernan el flujo de datos y funciones de procesamiento que tienen lugar en las diferentes unidades de la célula durante la ejecución de las primitivas. No obstante, por motivos de exposición, consideraremos asociadas a cada unidad los circuitos de generación de sus microórdenes específicas.

A fin de identificar el formato -implícito en la CCD- del tuple en procesamiento, la Sección de Control genera dinámicamente dos números binarios cuyos valores coinciden, en todo instante de tiempo, con el número de orden dentro del tuple que ocupan los bytes en fase de lectura y escritura.

A estos números los denominamos Campos Imaginarios de Lectura y Escritura, respectivamente, (CIL y CIE); sus valores irán desfasados en una cantidad igual a la longitud en bytes del Registro de Desplazamiento.

Antes de pasar al estudio de las diferentes unidades, hemos de reseñar que, por motivos de simplicidad en los esquemas, todas las zonas duplicadas del Espacio de Memoria de la célula que hacen posible el solapamiento de las funciones de carga y ejecución, se han dibujado una sola vez, indicándose con un doble recuadro su naturaleza (Figura III-4).

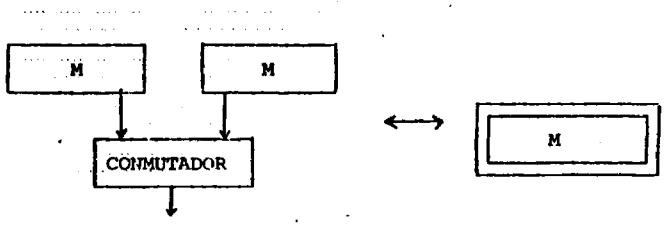


Figura III-4

III. 3.- Unidad de Cualificación: búsquedas por contexto

Sobre esta unidad recaen los procesos de búsqueda de la célula. Su capacidad selectiva de datos condiciona la eficiencia global del procesador en este tipo de funciones que son, por excelencia, las típicas de BD.

Aunque se ha propuesto el acceso por contenido en diferentes procesadores de BD ([5])([6]), es importante, además, que los datos puedan seleccionarse en el contexto de su estructura lógica. La direccionabilidad por contenido representa en los procesadores no-numéricos lo que

el direccionamiento directo en los de tipo Von Neumann ([7]). De la misma forma que este direccionamiento es esencial para tales procesadores pero su utilización resulta poco flexible en el acceso a estructuras de datos propias del tratamiento numérico como puedan ser, por ejemplo, los vectoriales, la direccionabilidad por contenido de los procesadores no-numéricos resulta inadecuada para expresar, en forma simple, determinados criterios de búsqueda fundamentales en los lenguajes de recuperación. Extendiendo la analogía, podemos decir que el acceso o búsqueda por contexto representa en los procesadores no-numéricos lo que los diferentes tipos de direccionamiento (relativo, indexado, etc.,) representan en los ordenadores convencionales.

La Unidad de Cualificación de las células del PCBD disponen de la capacidad de acceso por contenido y contexto. Esta última, traducida a la terminología relacional, y como vimos en el apartado II4.1.4, se reduce a la posibilidad de realizar, de forma directa, operaciones de θ-restricción. Esta unidad determinará, en el curso de un ciclo PCBD, qué tuples del elemento de memoria asociado cumplen una función booleana de la forma:

$$B \equiv \bigvee_i T_i \qquad \delta \qquad B \equiv \bigwedge_i T_i \qquad (1)$$

donde cada término T_i será una condición simple exigida al campo de marca (término de marca) o a los valores de dominio (término de datos). El primero permite encadenar funciones del tipo (1) y expresar así funciones booleanas de cualquier complejidad, ya que éstas serán siempre descomponibles en una disyunción (conjunción) de conjunciones (disyunciones). Los términos de datos, como quedó expuesto en el apartado II.2.3.1,

pueden ser de dos formas:

1.- $\mathcal{D}_i \; \theta \; CTE$; $\theta \in \{<, \leq, =, \geq, >\}$, permitiendo la comparación de un dominio con una constante externa: búsqueda por contenido.

2.- $\mathcal{D}_i \; \theta \; \mathcal{D}_j$; $\{i \neq j\}$, permitiendo la comparación de dos dominios de un mismo tuple: búsqueda por contexto.

Desde un punto de vista operativo, esta unidad realiza las siguientes funciones:

1.- Determinación de los tuples activos

2.- Evaluación del término de marca

3.- Evaluación de los términos de datos

4.- Evaluación de expresiones disyuntivas de términos de datos cuando la célula funciona como receptora en operaciones de intercambio.

Las tres primeras se realizan en forma "pipe-line" sobre las respectivas subunidades (Fig. III-5). La cuarta, al igual que la tercera, se realiza sobre el Evaluador de Términos de Datos pero con el auxilio de los Registros de Retención de Dominios.

Estudiaremos, a continuación, la estructura y funcionamiento de cada una de estas subunidades.

III.3.1.- Detector de tuples activos

La función de esta subunidad no es propiamente de cualificación: determina la existencia o no de un tuple activo en una posición de tuple de la CCD. Sin embargo, dicha determinación, al condicionar todo el proceso posterior de la célula, comenzando por la evaluación de

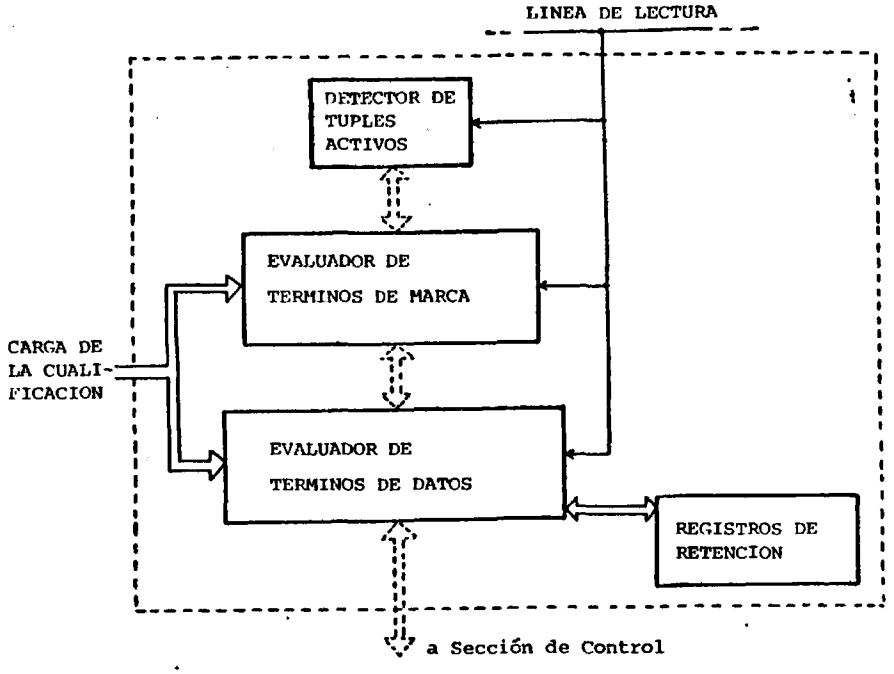


Figura III-5.- Unidad de Cualificación

la cualificación, la hemos incluido dentro de esta unidad.

Su estructura operativa la constituye un biestable síncrono D (Figura III-6) cuya entrada es la LINEA DE LECTURA, procedente de la CCD, y cuya salida de ACTIVACION indicará a la Sección de Control de la célula, durante el intervalo de lectura de la posición, la existencia o no de un tuple activo. Su carga se realiza con una señal CT, generada en la Sección de Control en sincronismo con la lectura del primer bit (bit A) de cada posición de tuple. La desactivación (puesta en baja del biestable) la realiza también la Sección de Control a través de la señal FT (fin de tuple) coincidiendo con la lectura del último bit.

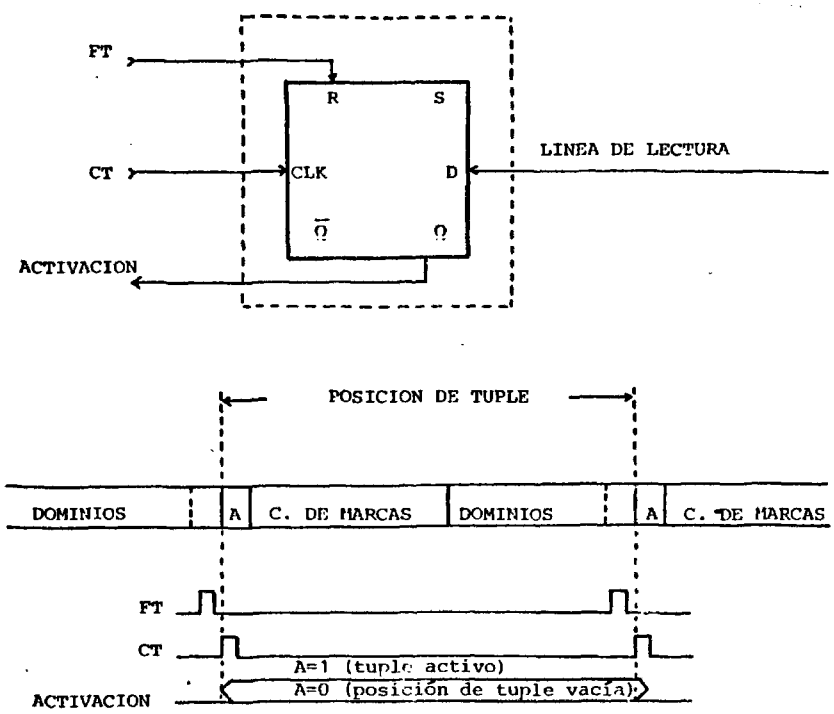


Figura III-6.- Detector de tuples activos y señales asociadas

III.3.2.- Evaluador de términos de marca

El término de marca T_m de una cualificación (ver apartado II.2.3.1) es una expresión booleana conjuntiva o disyuntiva simple de términos t_i $(i = 1,\ldots,4)$:

$$T_m = \bigwedge_{i=1}^{4} t_i \qquad \acute{o} \qquad T_m = \bigvee_{i=1}^{4} t_i$$

donde cada t_i es una comparación de un bit de marca m_i con un valor binario constante (1,0). No todos los bits de marca m_i han de formar parte de un término T_m.

A efectos operativos, un término T_m vendrá definido por:

1.- Una configuración binaria de cuatro bits, denominada <u>configuración valor</u>, con los valores binarios constantes de t_i en las posiciones correspondientes y los restantes irrelevantes.

2.- Una configuración binaria de cuatro bits, denominada <u>configuración máscara</u>, con valor "1" en posiciones correspondientes a los t_i que forman T_m, y "0" en los restantes.

3.- Un valor binario de la variable M(y/o) para fijar el carácter conjuntivo o disyuntivo de T_m.

Así, los términos $T_m \equiv (M = 1\ x\ o\ x)$ y $T'_m \equiv (M\ d\ 1\ o\ o\ x)$ vendrán definidos por:

$$\begin{cases} \text{c. valor} = 1\ 0\ 0\ 0 \\ \text{c. máscara} = 1\ 0\ 1\ 0 \\ \text{M(y/o)} = 1 \end{cases} \quad y \quad \begin{cases} \text{c. valor} = 1\ 0\ 0\ 0 \\ \text{c. máscara} = 1\ 1\ 1\ 0 \\ \text{m(y/o)} = 0 \end{cases}$$

Además de la expresión de T_m, la primitiva correspondiente especificará en <CM> el campo de marca concreto, de los N existentes, sobre el cual operará la cualificación.

La estructura operativa de esta subunidad se muestra en la Fig. III-7. Básicamente consta de los registros que mantienen los parámetros de definición de T_m (R. VALOR, R. MASCARA y R. CM), un Registro de MARCA donde se cargan secuencialmente y en forma serie los campos de marca, a medida que sus correspondientes tuples son procesados, y un Operador Booleano donde propiamente se realiza la evaluación.

Las variables $T\ M\ L_i$ $(i = 1,\ldots,N)$ cuya activación coincidirá con la presencia de los sucesivos campos de marca en la línea de lectura, se generan en la Sección de Control y permiten la carga en R. MARCA

del campo seleccionado en R.CM

Siendo v_i, m_i, M_i $(i = 1,\ldots,4)$ las variables de salida correspondientes a cada uno de los registros, la ecuación lógica del operador booleano será:

$$STM = y/0 \{ \bigwedge_{i=1}^{4} [\overline{m_i} + (\overline{v_i \oplus M_i})] \} + \overline{y/0} \{ \bigvee_{i=1}^{4} [\overline{m_i} + (\overline{v_i \oplus M_i})] \}$$

donde la variable y/o determinará el tipo conjuntivo o disyuntivo de T_m. La variable STM, cuyo estado indicará el cumplimiento o no del término de marca por el tuple en fase de procesamiento, será llevado a la Sección de Control.

III.3.3.- Evaluador de términos de datos

El número de términos que pueden entrar a formar parte en una expresión booleana de cualificación, evaluable en una revolución de la memoria rotante, constituye uno de los factores determinantes de la capacidad de procesamiento de los dispositivos de lógica distribuida. En efecto, este valor fijará el número de revoluciones necesarias para resolver un determinado acceso por contenido. En la mayoría de los procesadores propuestos ([8]) esta capacidad va ligada al número de comparadores disponibles en los elementos lógicos de las células. Así, RAP ([9]) sólo puede evaluar directamente cualificaciones por contenido con el número de términos igual al de comparadores. Evidentemente, esto supone una limitación a la hora de procesar relaciones de grado elevado con criterios selectivos que referencian un gran número de dominios.

En el PCBD hemos superado esta limitación diseñando el Evalua-

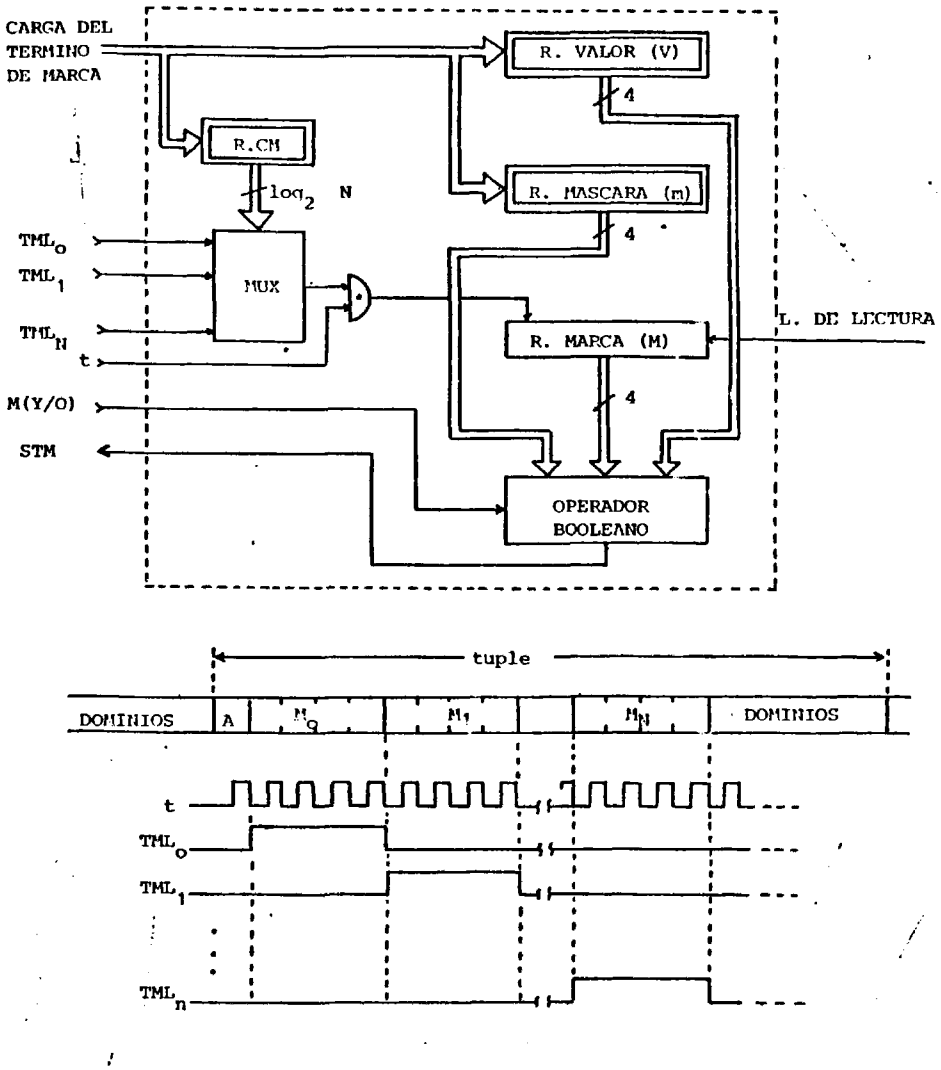


Figura III-7.- Evaluador de términos de marca y señales de control asociadas.

dor de Términos de Datos con capacidad para resolver directamente una cualificación conjuntiva o disyuntiva simple con tantos términos como dominios tenga la relación y utilizando un sólo comparador ([10]). Para ello, las constantes de comparación se llevan dinámicamente al comparador, a medida que los correspondientes dominios se leen de la memoria rotante.

Otro factor importante que contribuye a la capacidad selectiva de los dispositivos de lógica distribuida lo constituye la posibilidad de evaluar directamente criterios de búsqueda por contexto. Esta característica, puesta ya de manifiesto en el capítulo anterior, también ha sido incorporada al Evaluador de Términos de Marca.

Esquemáticamente, esta subunidad se compone de una memoria para soportar la expresión booleana de cualificación, y un comparador donde se efectúan las evaluaciones individuales de cada término, a medida que los dominios referenciados se leen de la CCD (Fig. III-8).

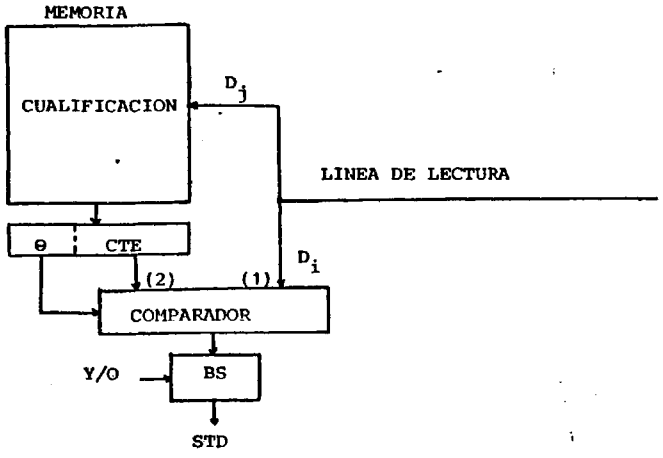


Figura III-8.- Esquema simplificado del Evaluador de Término de Datos

Cuando el término es de la forma $\mathcal{D}_i$ θ CTE (contenido), en sincronismo con la llegada a la entrada (1) del comparador, del valor de $\mathcal{D}_i$ procedente de la línea de lectura, la memoria posiciona a la entrada (2) la constante CTE correspondiente, así como el operador θ. Una vez efectuada la comparación, se memoriza el resultado en el biestable de salida BS, realizándose la "Y" u "O" lógica con los resultados de los términos anteriores, según sea el carácter de la cualificación (conjuntivo o disyuntivo).

Cuando el término es de la forma $\mathcal{D}_i$ θ $\mathcal{D}_j$ con $j < i$ (contexto), el valor del dominio que antes se lee de la CCD, en nuestro caso $\mathcal{D}_j$, se carga en la memoria para hacer el papel de constante CTE a la llegada del valor del segundo dominio $\mathcal{D}_i$. A partir de este punto, la evualuación tiene lugar en idéntica forma al caso anterior.

En el diseño de esta unidad, se podría utilizar, para soportar las constantes de la expresión booleana de cualificación, un registro de desplazamiento circular que rotase su contenido en sincronismo con la CCD ([8]). Sin embargo, el mantenimiento de la generalidad de la célula para relaciones de cualquier longitud (grado), obligaría a diseñarlo con longitud reconfigurable entre amplios valores (máximo y mínimo permitidos a los tuples) ocasionando una excesiva complejidad. Por otra parte, la utilización de una memoria tipo cola (FIFO) que, en principio parecería adecuada, limitaría el tamaño de la cualificación a la insuficiente capacidad de los que actualmente existen en el mercado, además, sería necesario que la cola fuese también accedida por dirección, para permitir la carga de los valores de dominio en términos del tipo $\mathcal{D}_i$ θ $\mathcal{D}_j$.

Los argumentos anteriores nos han llevado a utilizar una me-

moria RAM como soporte de las constantes y operandos de la expresión de cualificación (Fig. III-10).

Por cada dominio de nb bytes (nb comprendido entre 1 y 4) de los referenciados en la cualificación, se utiliza un bloque de $nb + 1$ palabras de la RAM para su codificación. Los diferentes bloques se almacenan secuencialmente a partir de la posición cero de memoria, siguiendo el orden que los respectivos dominios ocupan en la relación. La codificación de cada una de las palabras de un bloque será:

1.- Cuando el dominio pertenece a un término tipo $\mathcal{D}_i$ θ CTE; la primera codificará θ y el número de orden, en binario, que el primer byte del dominio $\mathcal{D}_i$ ocupa en el tuple; las nb palabras restantes codificarán la constante CTE.

2.- Cuando el dominio pertenece a un término del tipo $\mathcal{D}_i$ θ $\mathcal{D}_j$, la codificación de $\mathcal{D}_i$ será diferente a la de $\mathcal{D}_j$. Si $j < i$, esto es, el valor de $\mathcal{D}_j$ se lee antes que el de $\mathcal{D}_i$, la codificación respectiva será:

a) para $\mathcal{D}_i$; igual que en el caso anterior, haciendo la salvedad de que las nb palabras siguientes a la primera, se cargan dinámicamente durante el procesamiento de los tuples.

b) para $\mathcal{D}_j$; la primera palabra codifica el número de orden en el tuple de su primer byte, así como el carácter de primer dominio de un término $\mathcal{D}_i$ θ $\mathcal{D}_j$ que lo denotaremos con * en el mismo campo que ocupaba θ en el caso anterior; las nb restantes palabras almacenan las direcciones de las nb palabras de la RAM correspondientes al bloque que codifica $\mathcal{D}_i$.

Todas las palabras contienen dos bits de control, P y F (los más significativos). El primero diferencia a las palabras iniciales de bloque, denominadas de especificación, de los restantes, denominados de

valor. El segundo marca la palabra fin de cualificación.

En la Figura III-9, hemos representado la codificación en RAM (c) de una cualificación (b) con ambos tipos de términos (contenido y contexto) referenciando dominios de un tuple (a).

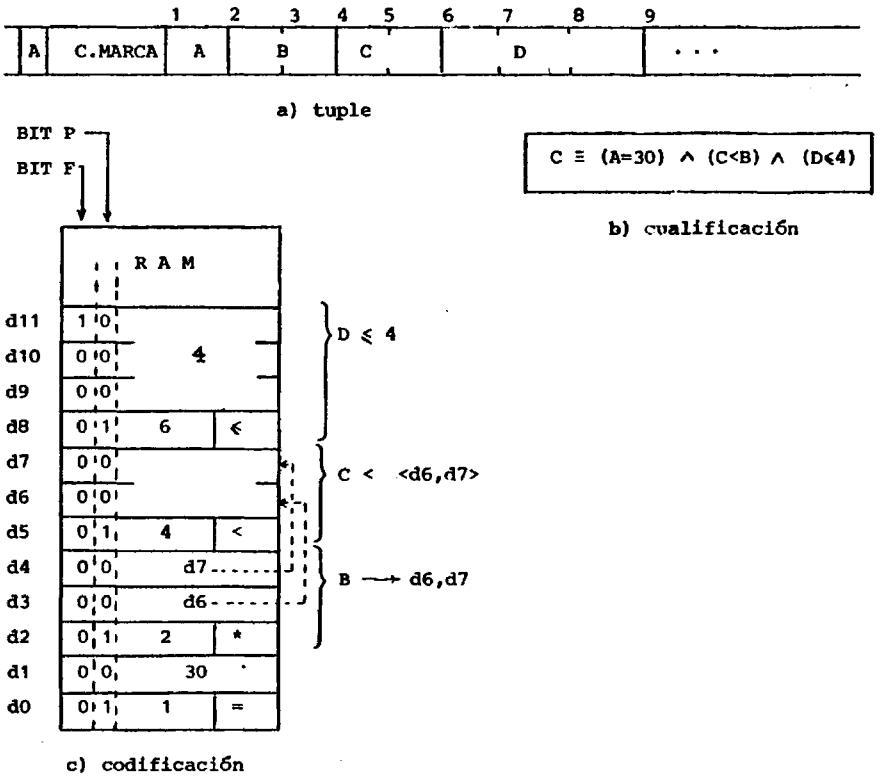


Figura III-9.- Codificación en RAM de los términos de datos de una cualificación.

La estructura completa de esta subunidad se muestra en la figura III-10.

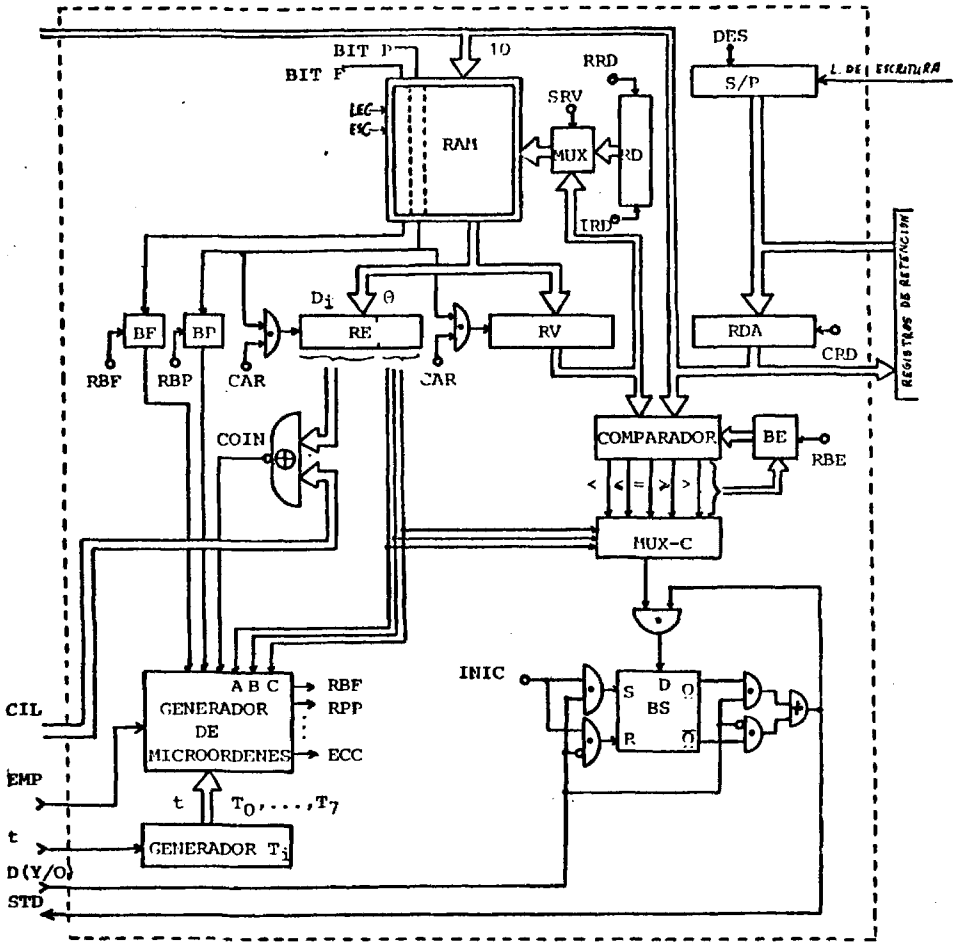


Figura III-10.- Estructura del Evualuador de Términos de Marca.

Su funcionamiento está siempre controlado por el contenido del registro de especificación (RE) y los biestables de fin (BF) y de palabra (BP). Su iniciación es responsabilidad de la Sección de Control (señal EMP).

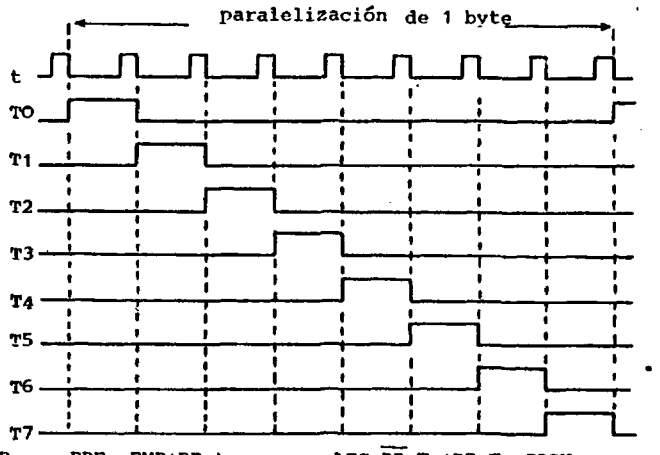
El registro de direcciones RD de la memoria RAM actúa en forma contador en la lectura consecutiva aunque no contigua de sus posiciones. Cuando la palabra que se lee es de especificación, el bit P hace que su carga se realice en RE; en caso contrario -cuando es de valor- la carga se efectúa en RV. Al mismo tiempo que los bits P y F posicionan los biestables BP y BF, respectivamente.

Cuando el contenido del campo D_i (número de orden del primer byte de un dominio referenciado en la cualificación) de RE, coincide con CIL (campo imaginario de lectura) la variable COIN = 1 desencadenará la comparación del byte paralelizado en S/P -previa transmisión al registro de datos RDA- con el primer byte de la CTE, leído de la memoria RAM sobre RV. El campo θ de RE selecciona, a través del multiplexor MUX-C, la línea de comparación correspondiente. El encadenamiento de las comparaciones parciales de los bytes que componen un dominio se efectúa en los biestables DE. El resultado final se almacena en BS.

El proceso anterior ocurre siempre que el campo θ corresponda un operador de comparación. Por el contrario, si dicho campo es *, significa que la palabra de especificación en RE pertenece a la codificación del primer dominio de un término de acceso por contexto. En este caso, el byte paralelizado en S/P y transmitido a RDA se escribe en la memoria RAM actuando RV como registro de direcciones. Este último registro se habrá cargado previamente con la dirección de la constante del segundo dominio del término por contexto.

El estado en alta de BF indica el fin del proceso de evaluación de un tuple. La variable de salida STD determinará su cumplimiento.

En la Figura III-11 aparecen las ecuaciones lógicas de todas las microórdenes que controlan el funcionamiento descrito de esta subunidad.



RBF=EMP;	RBE= EMP+BP.t_3;	LEC=$\overline{BP}$.T_1+BP.T_4.COIN
RBP=EMP;	RRD=BF.t_7;	CAR=$\overline{BP}$.t_1+BP.t_4.COIN
INIC=EMP:	ESC=SRV;	IRD=$\overline{BP}$.t_1+BP.t_4.COIN
DES=t;	CRD=t_o	SRV=A.B.C.(BP.COIN.T_5+$\overline{BP}$.T_2)

Figura III-11.- Ecuaciones lógicas de las microórdenes que gobiernan al Evaluador de Términos de Datos.

III. 3.4.- Registros de Retención

Cuando una célula ejecuta una primitiva de recepción, acepta valores de dominio DO_i a través del bus de Intercambio y forma con ellos, sobre la memoria RAM del Evaluador de Términos de Datos, cualificaciones disyuntivas de la forma:

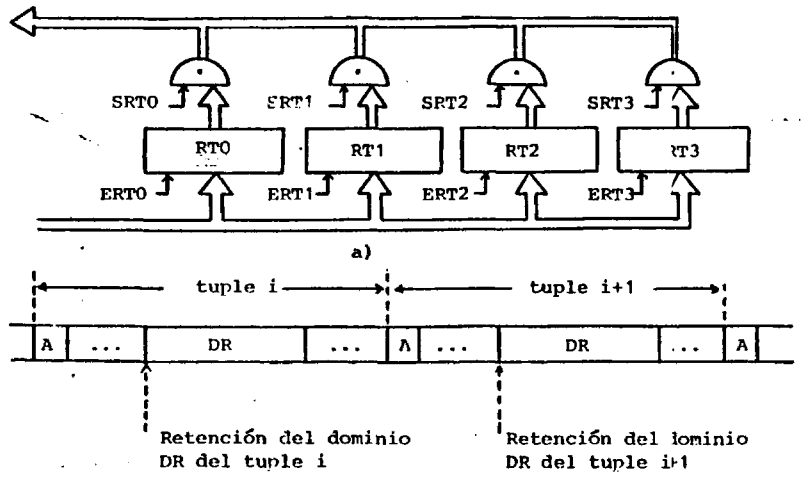
$$\bigvee_{i=1}^{p} (DR \ominus DO_i)$$

Esta función la realiza, como veremos en el apartado III.2.3, la Unidad de Intercambio. Sin embargo, su aplicación a los tuples del elemento de memoria de la célula es responsabilidad del Evaluador de Términos de

Datos.

Al tratarse de una cualificación cuyos términos referencian el mismo dominio DR, será necesario, para cada tuple, retener su valor al efectuarse la comparación del primer término. De esta forma, durante el tiempo que tarda la llegada del mismo dominio en el siguiente tuple, se podrán evaluar los sucesivos términos de la cualificación (Figura III-12,b). Evidentemente, si es ℓt la longitud en bytes de los tuples de la célula y ℓd la del dominio de DR, el número de términos p que se pueden evaluar en una revolución, deberá cumplir: $p \leq \ell t/\ell d$. Este valor es conocido por la U.C., y será transmitido como parámetro de funcionamiento a la Unidad de Intercambio en la primitiva de recepción.

Los registros que cumplen la función de retención, RTI (I = = o,1,2,3), se muestran en la Figura III-12,a). Su control (carga y salida) se realiza en forma análoga a la que estudiaremos para los registros de la Unidad Aritmética.



b)

Figura III-12

III. 4.- Unidad Aritmética

Sobre esta unidad se realizan todas las operaciones de tipo aritmético implícitas en las diferentes primitivas de célula, esto es, en las de modificación (suma, resta, sustitución) y agregación (máximo, mínimo, número-cantidad). Básicamente, consta de dos registros fuente (V y F), un operador aritmético (oA) y un registro destino (D) (Figura III-13).

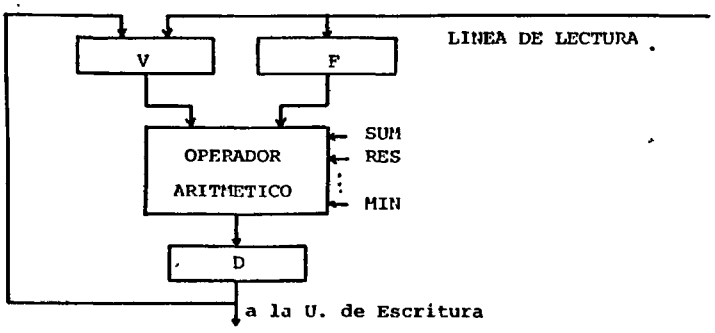


Figura III-13.- Estructura báscia de la Unidad Aritmética

En las operaciones de modificación, V, se carga con el operando modificador y F con el modificado, esto es, con el valor de dominio procedente del tuple en fase de procesamiento. Una vez realizada la operación correspondiente en oA, el resultado se carga en D. De aquí será transferido a la Unidad de Escritura donde sustituirá, en caso de cumplirse la cualificación, al valor de dominio leído de la CCD.

Cuando la modificación es por contexto, es decir, cuando el operando modificador es un valor de dominio del propio tuple, el registro V se carga análogamente a F. En caso contrario, la carga tiene lugar en la fase de inicialización de la célula.

En las funciones de agregación V contiene en todo momento el

valor actual de la función correspondiente. Inicialmente se carga con el menor número negativo, para la función máximo; con el mayor positivo, para la función mínimo; y con cero, para la función cantidad. Cada vez que el dominio referenciado de un tuple se identifica en la línea de lectura, se carga en el registro F. El operador oA, si se trata de la función cantidad, efectúa la suma de los contenidos de V y F, alevando el resultado a D. Para las funciones máximo y mínimo compara ambos contenidos, transfiriéndose el de F a D en el proceso. Si la comparación resulta cierta -sólo para las dos últimas funciones- y el tuple cumple la cualificación -para las tres funciones-, el contenido de D se transfiere a V, actualizándose así el valor de la función. Al finalizar un ciclo, todos los tuples habrán sido procesados y V contendrá el valor de la función de agregación correspondiente a los datos de la célula.

Debido a la longitud variable (de 1 a 4 bytes) de los dominios del PCBD, los registros V, F y D se componen, a su vez, de 4 registros independientes de 1 byte de longitud (Figura III-14). El Operador Aritmético funciona en forma byte-serie, encadenando las operaciones parciales y almacenando las condiciones finales en los biestables BC.

Supuesta inicializada la unidad con el operando modificador en V (si no es por contexto) y con los parámetros que definen el dominio o dominios referenciados NBT (número de orden que ocupa en el tuple el primer byte) y NB (número de bytes que componen el dominio), en el Generador de Microórdenes, las sucesivas fases de funcionamiento serán:

1) Fase de Carga.- Efectúa la detección y carga del dominio o dominios referenciados en los registros F y V. Es decir, cuando el valor de CIL (campo imaginario de lectura) coincide con uno de los dos parámetros NBT, comienza la operación de carga. Finaliza cuando el número

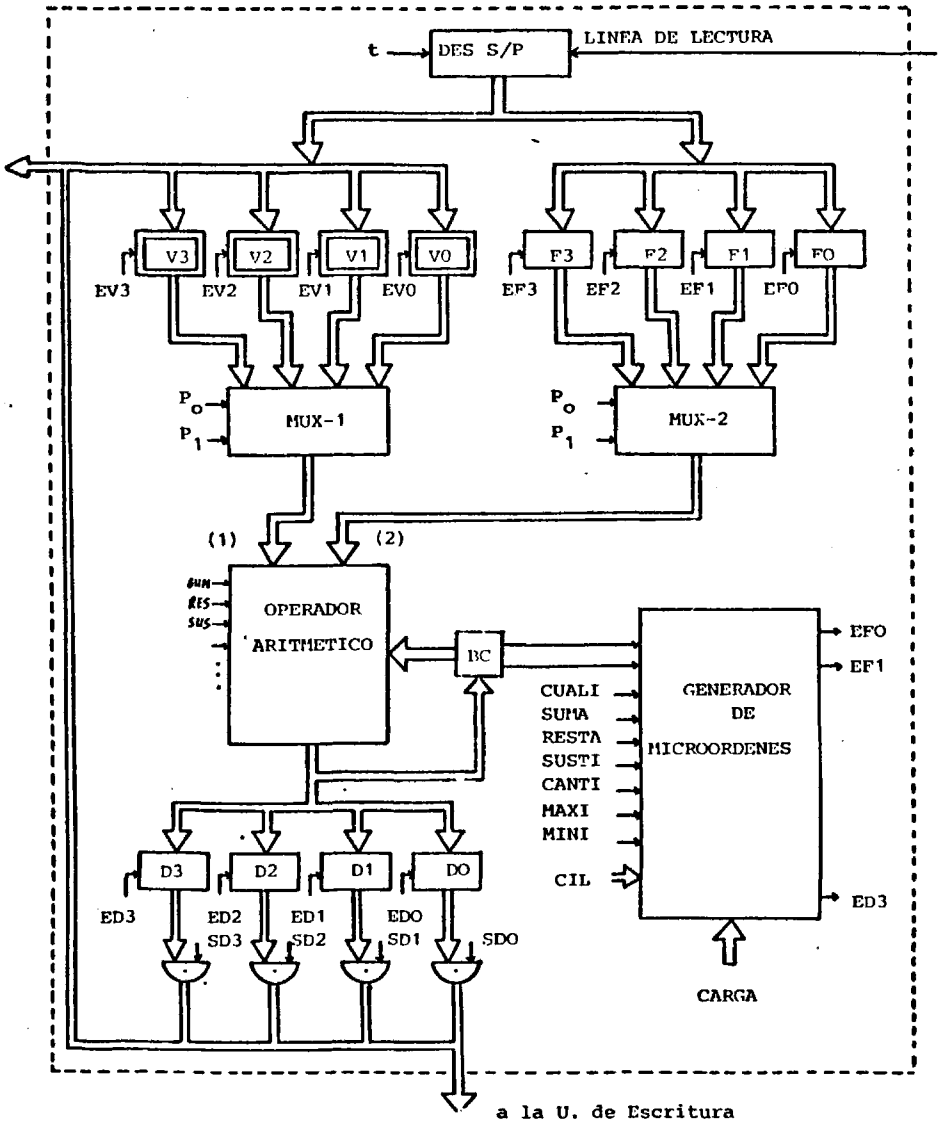


Figura III.4.- Estructura de la Unidad Aritmética

de bytes cargados sea igual a NB. En la figura III-15 se muestra uno de los dos circuitos del Generador de Microórdenes que posiciona la variable de CARGA.

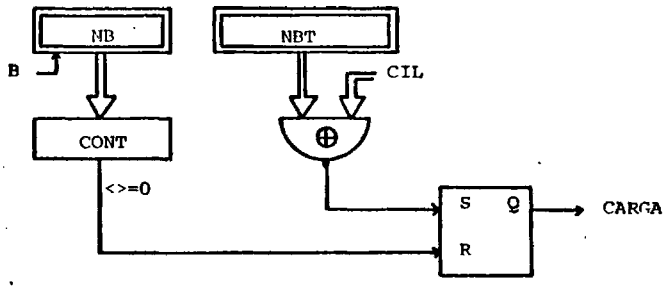
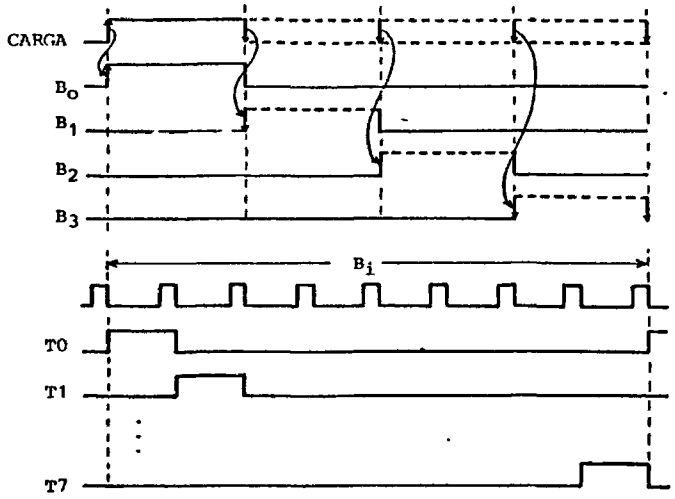


Figura III-15.

Las ecuaciones lógicas correspondientes a las microórdenes generadas en esta fase, así como las variables internas, se muestran en la figura III-16. Cada B_i $(i = 1,\ldots,4)$ es el tiempo de paralelización de un byte del dominio a cargar. Cuando se trata de una modificación por contexto existirán dos fases de carga, una para cada dominio

2) Fase de Operación.- Se efectúa cualquiera que sea la primitiva de agregación o modificación que se esté ejecutando. El número de intervalos T'_i de duración de esta fase, será igual al número de bytes del dominio operado, y vendrá controlado por el estado de una variable interna de operación (VOPERA) generada de forma análoga a la de carga, esto es, puesta en alta con la finalización de la fase de carga y en baja con la cuenta a cero de NB. Las ecuaciones lógicas de las microórdenes correspondientes a esta fase se muestran en la figura III-17. La duración de los intervalos T'_i los hemos hecho iguales a los de acceso (lectura y escritura) de un bit de la CCD.

3) Fase de actualización.- Para las primitivas de agregación

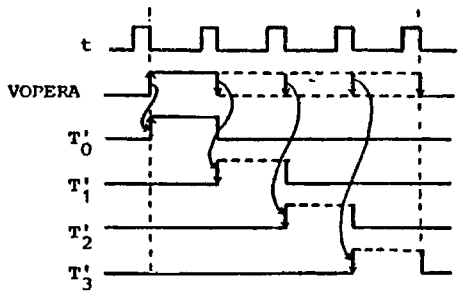


EFO=t_7.B0.CARGA(F); EVO=t_7.B0.CARGA (V)
EF1=t_7.B1.CARGA(F); EV1=t_7.B1.CARGA(V)
EF2=t_7.B2.CARGA(F); EV2=t_7.B2.CARGA(V)
EF3=t_7.B3.CARGA(F); EV3=t_7.B3.CARGA(V); t_1=t.T_i

Figura III-16.- Ecuaciones lógicas de las microórdenes generadas en la fase de carga.

esta fase se efectúa siempre que la comparación haya sido cierta (BC = 1) al igual que la cualificación del tuple (CUALI = 1). Consiste en la transferencia de los contenidos de D_i a F_i. Suponiendo que las variables T'_i se generan de forma análoga a la fase de operación, bajo control, en este caso, de la variable de actualización (VACTU) que a su vez estará condicionada a BC y CUALI, las ecuaciones de las microórdenes correspondientes serían:

salida del registro DO, SDO = T'_0 . VACTU
" " " D1, SD1 = T'_1 . VACTU
" " " D2, SD2 = T'_2 . VACTU
" " " D3, SD3 = T'_3 . VACTU



$P_o=(T'_1+T'_3)$.VOPERA; EDO=t'_o.VOPERA; ED3=t'_3.VOPERA
$P_1=(T'_2+T'_3)$.VOPERA; ED1=t'_1.VOPERA; $t'_1=t.T_i$
SUM,RES,PD1,COM=VOPERA; ED1=t'_2.VOPERA;

Figura III-17.- Ecuaciones lógicas de las microórdenes generadas en la fase de operación.

entrada en el registro V0, EVO = t'_o . VACTU
" " " " V1, EV1 = t'_1 . VACTU
" " " " V2, EV2 = t'_2 . VACTU
" " " " V3, EV3 = t'_3 . VACTU

En las primitivas de modificación, esta fase transfiere el contenido de los registros D_i a la Unidad de Escritura, siempre que el tuple haya satisfecho la cualificación (CUALI = 1). Las ecuaciones de las microórdenes correspondientes son idénticas a las anteriores si sustituimos EVI(I = o,1,2,3) por las variables de carga de los registros de la Unidad de Escritura.

III.5.- Unidad de Intercambio

Esta unidad nó es esencial desde un punto de vista conceptual.

Su presencia en la célula obedece a motivos de rendimiento. Permite la transmisión directa de datos entre dos grupos de células para extender dinámicamente criterios de búsqueda, con uniones implícitas en su formulación, a través de varias relaciones.

Sus dos modos de funcionamiento: transmisión y recepción, responden a las exigencias operativas de las primitivas del mismo nombre; a través de las cuales la U.C. ejecuta la instrucción de Selección Indirecta.

En el modo de transmisión, para cada tuple en fase de escritura, esta unidad detecta y deriva hacia el bus de Intercambio el dominio de transmisión especificado en la primitiva, cuando en la fase de lectura cumplió la cualificación (Figura III-18a). La utilización del bus de Intercambio es controlado por un dispositivo de Selección por Prioridad Encadenada (Daisy chaining) asociado a dicho bus para arbitrar su asignación cuando se dan demandas simultáneas.

En el modo de recpeción acepta valores de dominio procedentes del bus de Intercambio con los que construye, sobre la memoria RAM del Evaluador de Términos de Datos, cualificaciones disyuntivas del tipo visto en la correspondiente primitiva (Figura III-18,b).

Estudiaremos, independientemente, los dos modos de funcionamiento de esta unidad, así como la estructura operacional asociada a cada uno de ellos.

- Modo Transmisión

En este modo de funcionamiento, la unidad de Intercambio es inicializada con los parámetros que definen, dentro de un tuple, el posible dominio a transmitir:

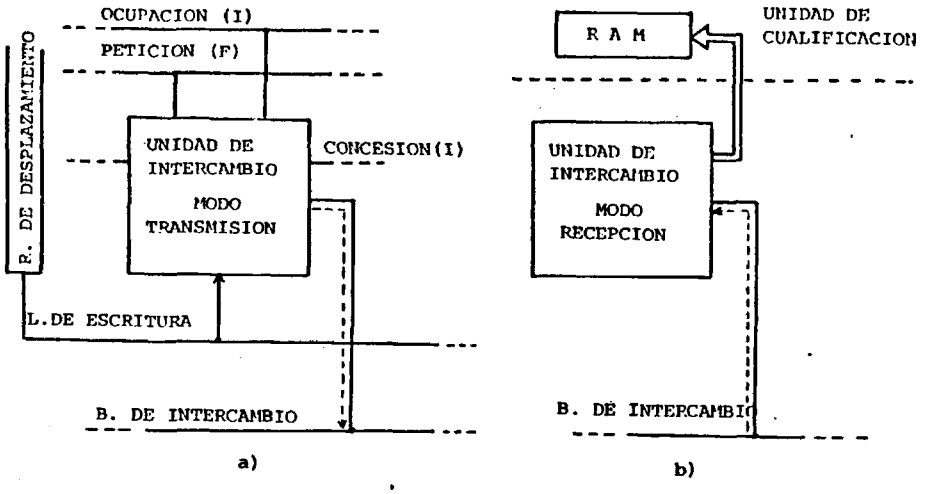


Figura III-18

NBT: Número de orden que ocupa en el tuple el primer Byte del dominio a Transmitir.

NB: Número de Bytes que componen el dominio.

Estos valores son cargados en los registros correspondientes de la unidad (Figura III-19). El "driver" que controla la direccionalidad de la transmisión es también inicializado (T/R = 1).

Si la cualificación ha sido satisfecha por un tuple y el bus de Intercambio está desocupado (BO = 1), se transmitirá el correspondiente dominio tan pronto entre en fase de escritura. Ello ocurrirá cuando el valor del Campo Imaginario de Escritura (CIE) coincida con el contenido del registro NBT. Esta coincidencia desencadenará, pues, el comienzo de transmisión (COT = 1) actuando sobre la entrada S del biestable BT.

El tiempo que dura la transmisión vendrá determinado por el

número de bytes que componen el dominio. Su finalización la marcará la señal FOT generada cuando el contenido del contador CON, cargado inicialmente con <NB>, sea igual a cero. Este contador se decrementa a intervalos bytes (B) a partir del comienzo de transmisión. La señal FOT actuará sobre la entrada R de BT.

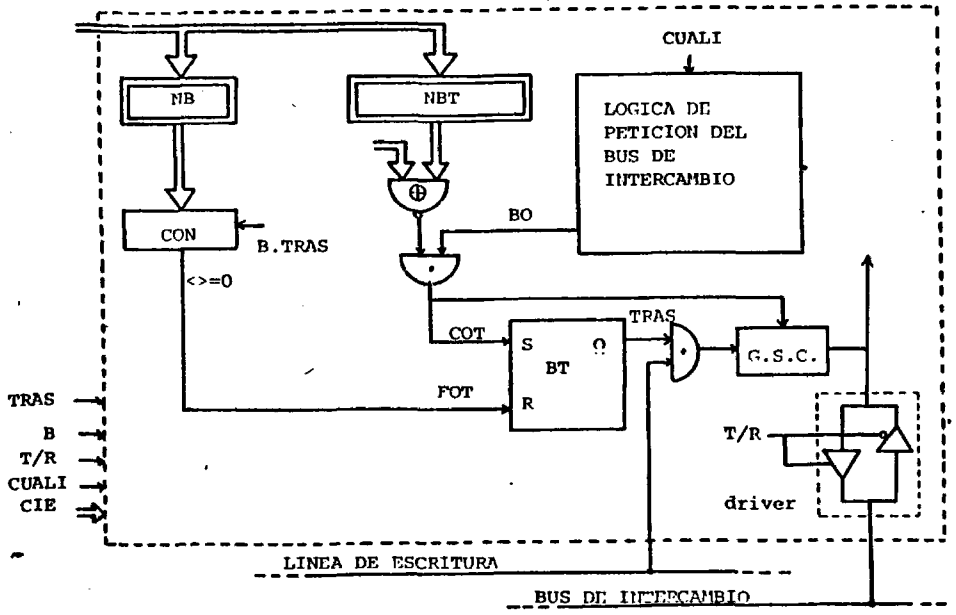
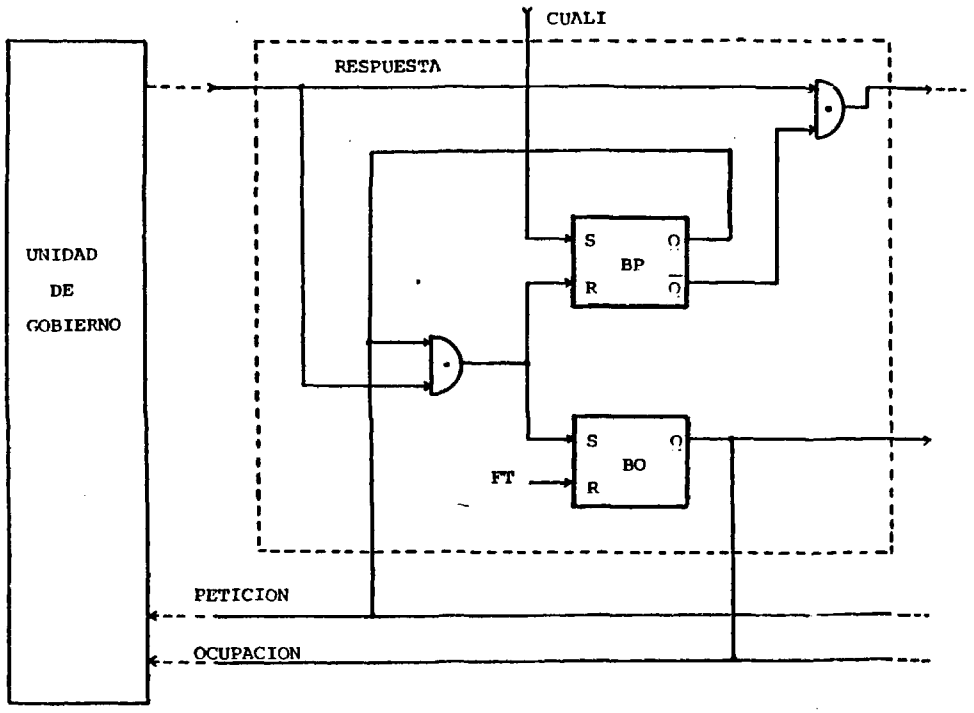


Figura III.19.- Unidad de Intercambio: modo transmisión

Previa a la transmisión del dominio, el Generador de Señal de Comienzo sincronizará la célula transmisora con el conjunto de células receptoras. Esta señal puede estar compuesta simplemente por un bit "1".

La lógica de Petición del bus de Intercambio la componen dos biestables: Biestable de Petición (BP) y Biestable de Ocupación (BO), (Figura III-20), conectados a través de las líneas de PETICION, OCUPA-

CION y RESPUESTA a la Unidad de Gobierno correspondiente en la U.C.



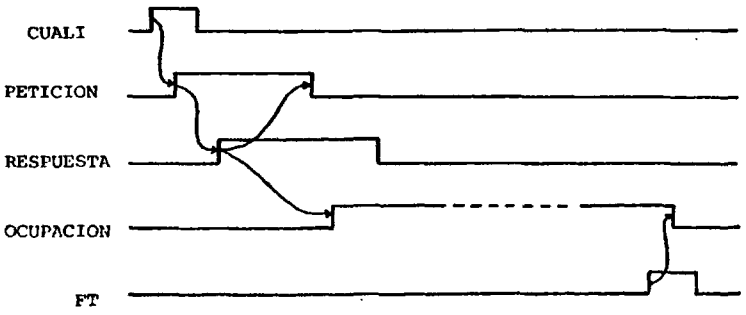


Figura III-20.- Lógica de Petición del bus de Intercambio y señales de control asociadas.

Si la cualificación ha sido satisfecha por un tuple (CUALI=1), el biestable BP pasará al estado "1" y activará la línea de PETICION al tiempo que desconecta la línea de RESPUESTA de todas las células a su derecha. La Unidad de Gobierno responderá, generando una señal por la línea de RESPUESTA, siempre que la de OCUPACION esté desactivada. Esta señal llegará a la célula si no ha existido a su izquierda alguna otra que, simultáneamente, hiciera la petición y desconectara la línea de RESPUESTA. La recepción de la señal de respuesta será, pues, quien determine la utilización del bus. Esta pondrá a "O" BP, al tiempo que activa la línea de OCUPACION (BO = 1).

- Modo Recepción

En este modo de funcionamiento, la Unidad Intercambio construye, en la memoria RAM del Evaluador de términos de Datos, cualificaciones disyuntivas de la forma:

$$CD \equiv \bigvee_{i=1}^{p} (DR \ \theta \ OP_i)$$

con los valores OP_i recibidos por el bus de Intercambio.

La estructura operacional de esta unidad para funcionar en modo recepción se muestra en la Figura III-21. Tres son los registros inicializados por la primitiva de recepción con los parámetros correspondientes a la definición de CD:

RP - almacena la palabra de especificación de todos los términos de la cualificación, esto es:

0	1	DR	θ

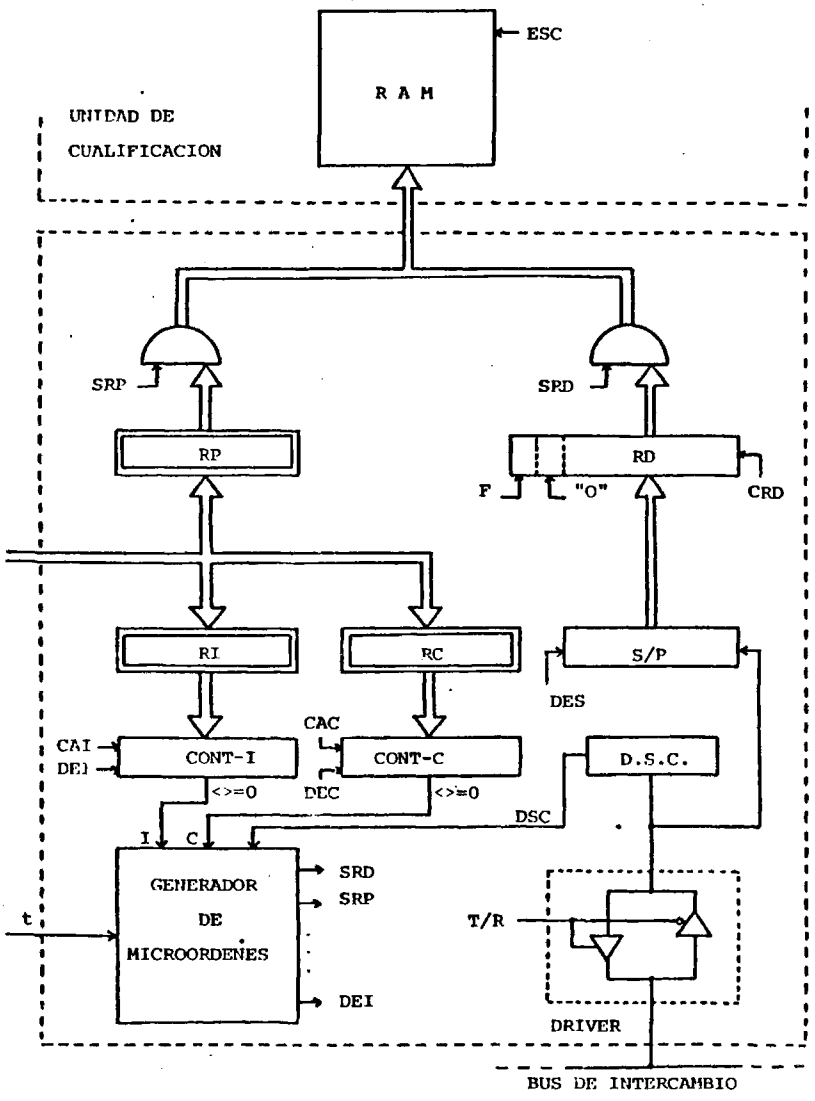


Figura III.21.- Unidad de Intercambio: modo recepción

RI - almacena el número de bytes del dominio DR

RC - almacena el número máximo p de valores OP_i que la unidad acepta en una revolución.

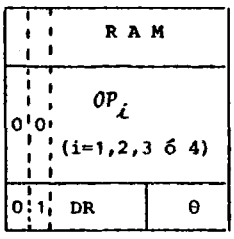
Cada vez que el Detector de Señal de Comienzo (D.S.C.) identifica una señal de este tipo en el bus de Intercambio, por otra célula funcionando en el modo transmisión, se desencadena el proceso de formación de un término $DR \ \theta \ OP_i$ en la memoria RAM.

El proceso consistirá en el almacenamiento secuencial en la RAM de:

1.- el contenido de RD

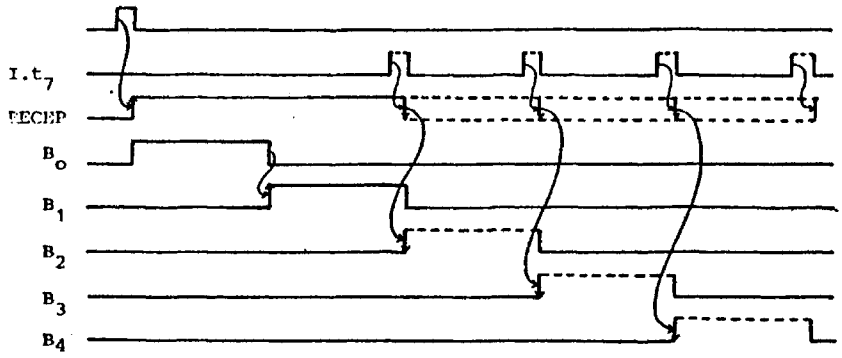
2.- los bytes siguientes a la señal detectada por D.S.C., previamente paralelizados en S/P y transmitidos a RD.

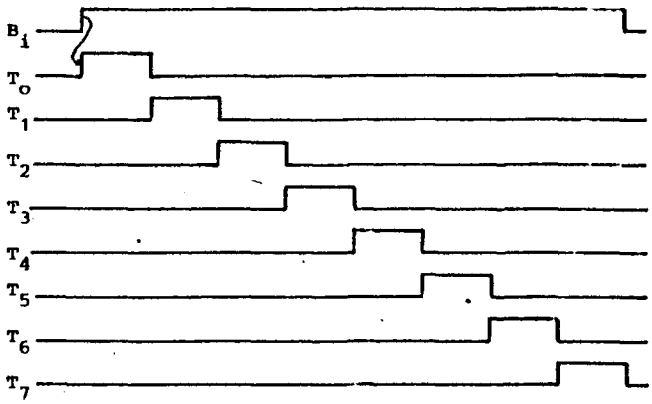
El resultado será la formación de un término codificado de la forma:



Los bits más significativos del registro RD, correspondientes a los de control de las palabras codificadas, están cableados por hardware. El primero permanentemente a "0", como corresponde a las palabras de valor. El segundo controlado por la variable F para marcar a "1" la palabra fin de cualificación.

En la Figura III-22 se muestran las ecuaciones de las micro-





CARC= comienzo de ciclo PCBD

DEI=$t_7.\overline{B_o}$

CAI=$t_7.B_o$

SRP=$(T_o+T_1).RECEP.\overline{B_o}$

SRD=$(T_o+T_1).RECEP.B_o$

CRD = t_o.RECEP

DEC = $t_o.\overline{B_o}$

DES = RECEP . t

ESC = $T_1.\overline{B_o}$

Figura III-22

órdenes correspondientes a este modo de funcionamiento, así como los diagramas de tiempo de las variables internas que definen estas ecuaciones.

La variable C que identifica el final del proceso de recepción en una revolución, por haberse recibido el número máximo p de valores de dominio, actuará sobre una línea de control que recorre todas las células y cuyo estado impedirá que las emisoras envíen más dominios al bus de Intercambio durante dicha revolución.

III.6.- Unidad de Salida

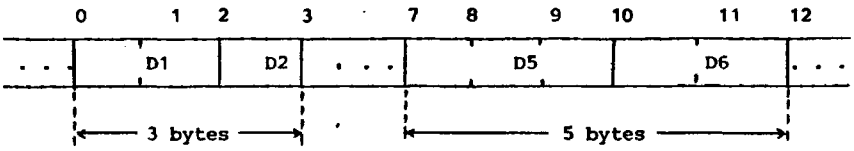
Todos los datos seleccionados en una célula, mediante los procesos de búsqueda implícitos en la ejecución de una secuencia de primitivas, son transmitidos a la U.C. a través de la Unidad de Salida.

Funcionalmente, esta unidad es análoga a la de intercambio en modo transmisión. Es decir, deriva hacia el bus de salida determinados dominios de aquellos tuples que cumplen una cualificación. Los dominios y expresión de cualificación se especifican en la primitiva <SA>, única del repertorio que utiliza esta unidad. El problema de la demanda simultánea del bus de salida por células soportando la misma relación y con tuples cualificados en idénticas posiciones angulares de la CCD, se resuelve de la misma forma que en la Unidad de Intercambio, esto es, mediante una lógica de petición del bus con prioridad encadenada (daisy-chaining).

En esta unidad, sin embargo, el número de dominios de un mismo tuple que se transmiten hacia el bus de salida varía de uno al grado de la relación soportada por la célula. Para evitar la redundancia que supondría la especificación individual de todos los dominios referenciados, para esta unidad un par de parámetros, NBT y NB, definirán no a un solo dominio sino a un conjunto consecutivo y contiguo de ellos. De esta

forma, NBT será igual al número de orden que el primer byte del primer dominio de un conjunto consecutivo, ocupa en el tuple; NB indicará el número total de bytes del conjunto.

Así, si la primitiva: <SA> <C> <CM> <DM> <D1, D2, D5, D6> se ejecuta sobre una célula cuyo formato de datos es:

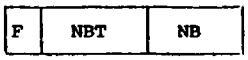


la Unidad de Salida será inicializada con los siguientes pares de parámetros:

$NBT_1 = 0$; $NB_1 = 3$ para los dominios D1 y D2

$NBT_2 = 7$; $NB_2 = 5$ para los dominios D5 y D6

La estructura operacional de esta unidad se muestra en la Figura III.23. Sobre las posiciones de una memoria RAM se almacenan, consecutivamente, los pares de parámetros que definen los diferentes conjuntos de dominios contiguos referenciados en la primitiva. El formato de cada palabra es:



El bit más significativo (F) marcará la última de la serie que define la totalidad de los dominios referenciados en <SA>. Sobre la RAM de la Figura III.23 hemos representado la serie de palabras correspondientes a los dominios especificados en el anterior ejemplo.

Con esta codificación, el caso más desfavorable, respecto al

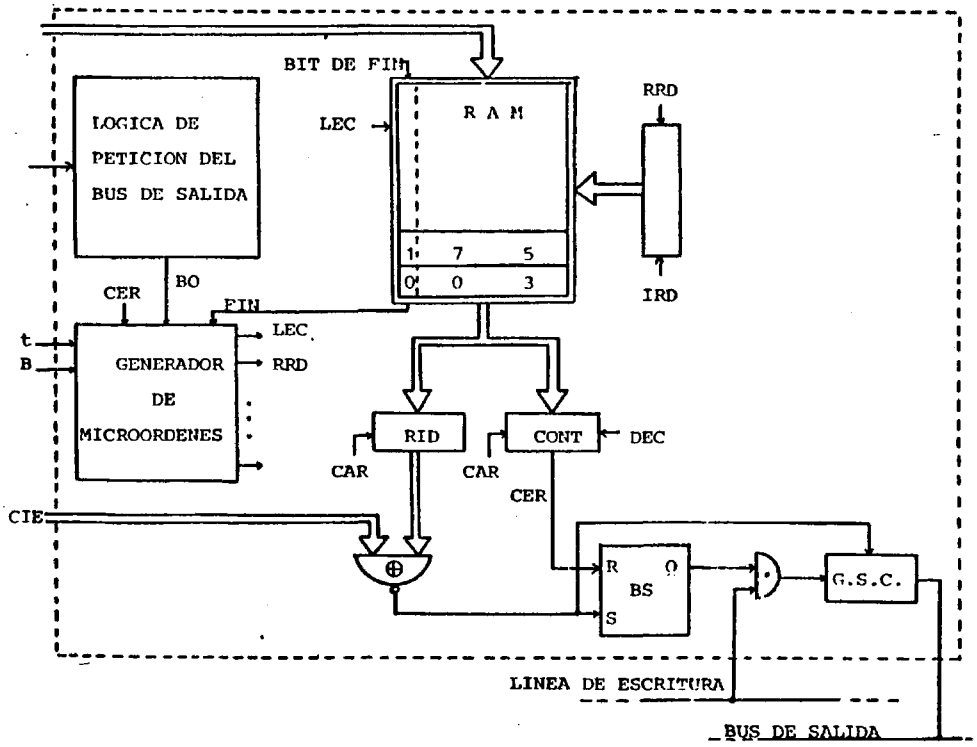


Figura III-23.- Estructura de la Unidad de Salida

número de palabras utilizadas, ocurrirá cuando la mitad de los dominios de una relación sean referenciados en <SA> y además éstos sean alternos. En este supuesto, el número de palabras será igual a la mitad del grado de la relación. Cuando todos los dominios aparezcan en la primitiva, bastará para su especificación con una sola palabra.

Cada vez que una célula procesa un tuple que cumple la cualificación, la Lógica de Petición del Bus de Salida (Figura III.23), idéntica a la descrita en al apartado III.2.3, solicita de la Unidad de Gobier-

no la utilización del bus. Si dicha célula recibe la señal de Concesión, procede a realizar la salida. En caso contrario, volverá a intentarlo en la siguiente revolución. En el primer supuesto, la señal BO desencadenará la lectura de la primera palabra de la memoria RAM. El contenido de sus campos NBT y NB se transfieren, respectivamente, al registro RD y al contador CONT. Cuando el valor del CIE coincida con el contenido de RID, el biestable de salida, BS, se activa y la información, en la línea de escritura, se deriva hacia el bus de salida. Previamente el Generador de Señal de Comienzo (G.S.C.), con idéntica función que la descrita en la Unidad de Intercambio, sincroniza la transferencia. Esta finaliza, temporalmente, cuando el contenido de CONT, cuyo decremento tiene lugar a intervalos byte, sea igual a cero (CER = 1). Cuando ello ocurre se procede a la lectura de la siguiente palabra de la RAM, repitiéndose un proceso análogo para el correspondiente conjunto de dominio contiguos. Cuando la palabra leída sea la última de la serie (bit F = 1) el proceso total de salida del tuple finaliza.

El Generador de Microórdenes de esta unidad es el encargado de distribuir las diferentes órdenes de gobierno, sobre los puntos de control de la estructura, a fin de llevar a efecto el proceso descrito.

III. 7.- Unidad de Escritura

Todas las modificaciones que una posición de tuple experimenta en la ejecución de las diferentes primitivas, se hacen efectivas en la Unidad de Escritura. La función de esta unidad es, pues, de la sustituir parte de la información serie de un tuple leído de la CCD por otra, elaborada en fase de lectura, antes de proceder de nuevo a su reescritura.

Aunque esta unidad participa en la ejecución de casi todas las primitivas, sus funciones pueden dividirse en tres:

1.- Sustitución del campo de marcas.

2.- Sustitución de dominios.

3.- Inserción y activación de tuples nuevos.

Cada una de estas funciones se corresponden con partes bien diferenciadas de su estructura operacional mostrada en la Figura III.24. Su estudio lo haremos separadamente.

1.- La sustitución de un campo de marca, CM, por la configuración especificada en la descripción de marca DM de una primitiva, se realiza siempre que un tuple cumple la cualificación C. Además, para las primitivas de Salida (SA), Transmisión (TR) y Recuperación (RE) será necesario la recepción de la correspondiente señal de Concesión.

Teniendo en cuenta que una descripción de marca no referencia necesariamente todos los bits del campo, su definición necesita, análogamente al término de marca de una cualificación, una configuración valor (V) y otra máscara (M). Si m es la configuración binaria del campo CM, será, pues, preciso realizar sobre él la operación booleana:

$$S_i = m_i V_i + \overline{m_i} M_i \qquad (i = 1,2,3,4) \qquad (1)$$

para no alterar los bits no referenciados en DM.

Está operación se realiza en fase de lectura simultánea y análogamente a la evaluación del término de marca. El resultado se carga en el registro de desplazamiento RM de la Unidad de Escritura (Figura III-24) para sustituir a CM en caso de cumplirse las condiciones oportunas. La localización de los registros M y V, soportes de DM, así como el Operador Booleano correspondiente a la ecuación lógica (1), podría haberse consi-

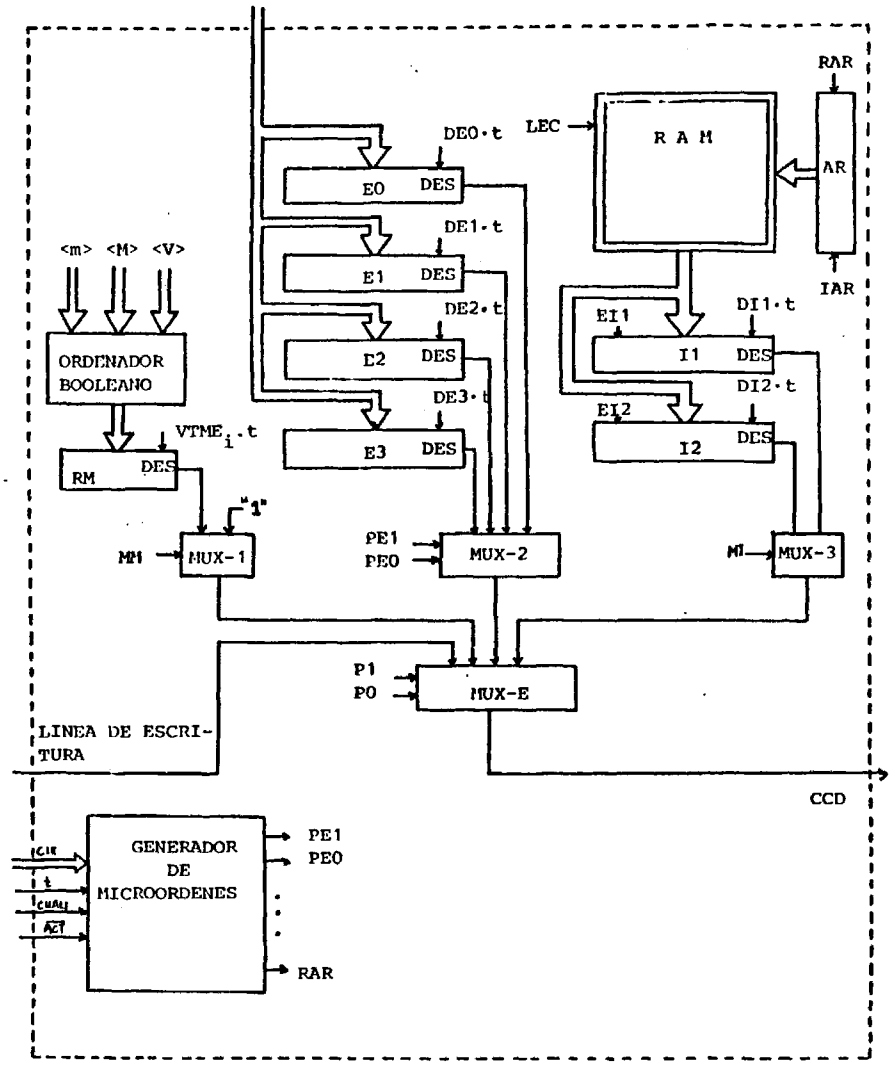


Figura III-24.- Estructura de la Unidad de Escritura

derado en la Unidad Aritmética; sin embargo, teniendo en cuenta el carácter específico de esta operación, hemos optado por englobarla en la Unidad de Escritura.

La sustitución del campo CM por el contenido de RM se realiza por desplazamiento serie (DES) de este último en coincidencia con la correspondiente señal TME_i, análoga a la TML_i, pero en fase de escritura. Además, la señal de peso MM del multiplexor MUX-1 y las p_0, p_1 de MUX-E mantendrán, durante la permanencia activa de TME_i, los valores:

$$MM = 0$$

$$p_0 = 0$$

$$p_1 = 0$$

a fin de permitir la conexión de la salida serie de RM con el punto de escritura de la CCD.

2.- La sustitución de dominios en la Unidad de Escritura sólo tiene lugar en las primitivas de modificación. En este caso, el valor del dominio especificado de un tuple, operado en la Unidad Aritmética y transmitido a los registros EI(I = 0,1,2,3) (Figura III-24) en fase de lectura, sustituirá al valor leído de la CCD, en fase de escritura, si la cualificación fue satisfecha por el citado tuple.

Al ser los dominios discretamente variables de 1 a 4 bytes, el número de registros EI que en cada caso haya que serializar y escribir sobre la CCD será igual al número de bytes del dominio modificado. Las variables DEI (I = 0,1,2,3), generadas secuencialmente a intervalos byte por la variable INMO (Figura III-25), determinarán dicho número. El estado en alta de la variable INMOD fija el tiempo que dura la sustitución. Dicha variable se genera a partir de los parámetros NBT y NB, que definen

el dominio modificado y el Campo Imaginario de Escritura (CIE).

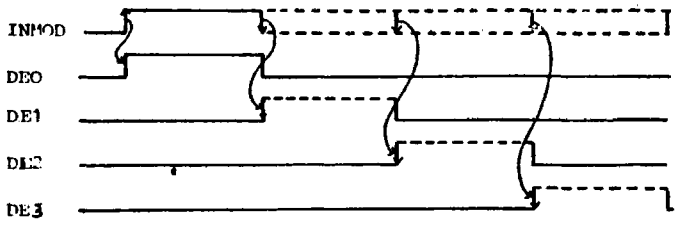


Figura III-25

Para conectar, durante el tiempo de sustitución, la salida serie de los registros EI(I = 0,1,2,3) con el punto de escritura de la CCD, las ecuaciones lógicas de las variables de peso de los multiplexores MUX-2 y MUX-E serán:

$$PEO = DEO + DE1$$

$$PE1 = DEO + DE2$$

$$PO = INMOD$$

$$P1 = 0$$

3.- Las operaciones de inserción de nuevos tuples en el elemento de memoria CCD de una célula, esto es, la primitiva <IN> se lleva a efecto en la Unidad de Escritura.

En la fase de carga de dicha primitiva, los nuevos tuples se almacenan, byte a byte, sobre una memoria RAM (Figura III-25). En fase de ejecución, la memoria RAM es leída secuencialmente y sus palabras serializadas y escritas en la CCD. Para ello, dispone de dos registros de datos asociados (I1, I2). Alternativamente, mientras el contenido de uno

se serializa, el otro recibe la siguiente palabra de la memoria RAM.

El estado en alta de la variable INSER, generada en la Sección de Control de la célula, determina el tiempo de inserción. Las variables DI1 y DI2, generadas a partir de INSER, según el diagrama de tiempo de la Figura III-26 efectúa, en coincidencia con la base de tiempo t, el desplazamiento alternativo de los registros DI.

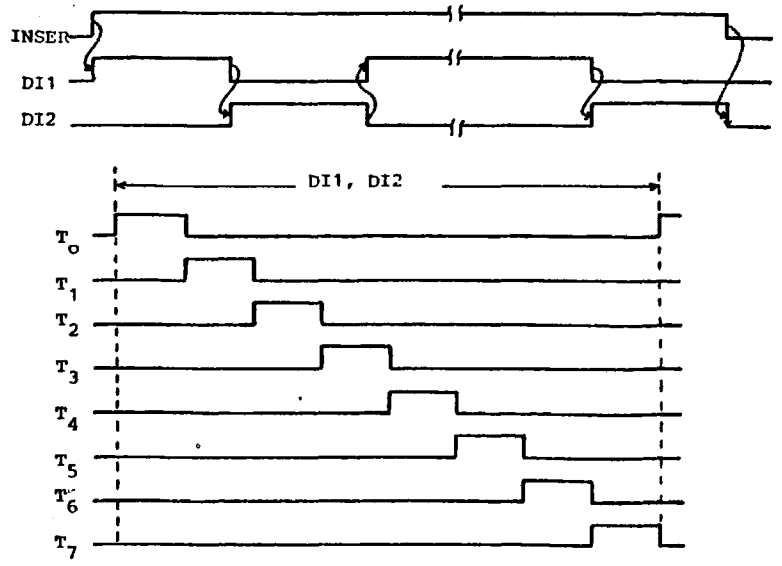


Figura III-26

La inserción tiene lugar sobre las posiciones de tuple desactivas (bit A = 0) detectadas en fase de lectura por la Unidad de Cualificación. Previa a la inserción, el bit A es activado. Para ello, las variables de peso de los multiplexores MUX-1 y MUX-E, en coincidencia con una señal que marca la presencia del bit A en la línea de escritura, tomarán los valores:

MM = 1

P0 = 1

P1 = 0

Durante la inserción, los multiplexores implicados en la conexión serán MUX-3 y MUX-E. Las ecuaciones lógicas de sus entradas de peso, así como la de todas las microórdenes que intervienen en la inserción, en función de las variables T_i de la Figura III-26, serán:

MI = DI1

P0 = INSER

P1 = INSER

LEC = T_0 . INSER

IAR = t_1 . INSER

EI1 = t_0 . DI1 . INSER

EI2 = t_0 . DI2 . INSER

III. 8.- COMUNICACION DE LAS CELULAS CON LA UNIDAD DE COORDINACION

Desde la U.C. una célula es un dispositivo periférico controlado a través de un Espacio de Memoria de acceso directo (RAM) (Figura III-27). Cada posición tiene una función específica en la ejecución de las diferentes primitivas. El desdoblamiento de este espacio en dos zonas idénticas para hacer posible el funcionamiento solapado de carga/ejecución de primitivas, no afecta a la referenciación unívoca de las diferentes posiciones: un sistema de conmutación hace que en cualquier instante de tiempo una tan solo de las zonas sea accesible desde la U.C.

El acceso se realiza por medio de dos buses: DIRECCIONES y DATOS. El primero selecciona la célula y posición específica de ésta,

implicada en la transferencia. A través del segundo se realiza la transferencia propiamente dicha.

Una célula puede seleccionarse de dos formas: absoluta y conjunta. Para la primera, cada célula tiene un código diferente que la identifica unívocamente. En forma conjunta, una célula se identifica cuando el código de selección, enviado por el bus de direcciones, coincide con uno previamente asignado por software.

Las 16 líneas del bus de direcciones $(a_0,\ldots,a_{15})$ forman tres grupos:

- Grupo 1.- Lo constituyen las 8 primeras líneas $(a_0,\ldots,a_7)$ y permiten referenciar, dentro de cada célula, una cualquiera de las 2^8 posiciones de su Espacio de Memoria.

- Grupo 2.- Lo forman las 7 líneas siguientes $(a_8,\ldots,a_{14})$ y permiten la selección, en forma absoluta, de una cualquiera de las 2^7 células que, como máximo pueden conectarse a la UC. El código de selección para cada célula lo determina la puerta de selección (Figura III-27). En forma conjunta, el código de selección viene determinado por el contenido del Registro de Relación, que constituye una posición más del Espacio de Memoria y sobre el que se puede escribir en forma absoluta.

- Grupo 3.- Lo constituye la línea más significativa (a_{15}) y su estado determina la selección absoluta o conjunta de las células.

Asociando un código diferente a cada relación y cargándolo en todos los Registros de Relación de las células que la soportan, el modo de selección conjunta posibilitará la carga simultánea sobre ellas de la misma primitiva. De esta forma, el número de procesos de carga que la UC realiza en cada ciclo PCBD será igual al número de relaciones, independientemente del número de células que las componen.

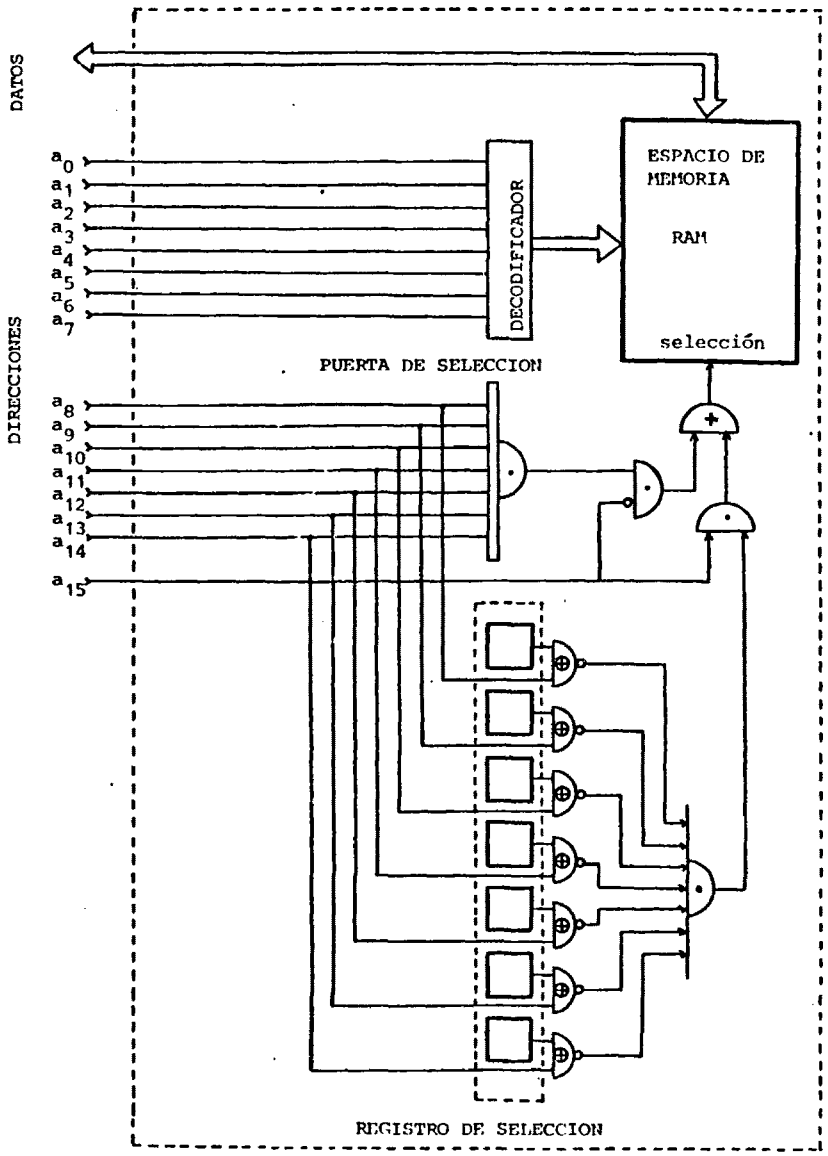


Figura III-27

III.9.- UNIDAD DE COORDINACION: EJECUCION DEL PROGRAMA OPERATIVO

Como dijimos en el Capítulo II, las funciones de la Unidad de Coordinación se soportan sobre un procesador de propósito general auxiliado por un "buffer" en el proceso de recepción de datos seleccionados en las células (Controlador de Salida). Por consiguiente, la estructura hardware de esta unidad no presenta ninguna singularidad. Sin embargo, sí es conveniente poner de manifiesto las exigencias temporales de ejecución del Programa Operativo respecto a la rotación de la información en los elementos de memoria CCD. Para ello, es conveniente diferenciar los siguientes tiempos:

- Tiempo de rotación de la CCD (tr), es el tiempo necesario para que un bit recorra toda la longitud de la memoria CCD.

- Tiempo de desplazamiento (td), es el tiempo necesario para que un bit recorra la longitud del Registro de Desplazamiento de una célula.

- Tiempo de ciclo PCBD (tc), es el necesario para que un bit recorra consecutivamente el Registro de Desplazamiento y la memoria CCD, es decir: tc = td + tr.

En la Figura III-28 hemos representado un diagrama correspondiente a estos tres tiempos, así como la localización de los datos de una célula (zona rayada de la Figura III-28) en los puntos A y B.

Es evidente que transcurrido el tiempo tr (punto A en el diagrama de la Figura III-28) todos los tuples habrán pasado la fase de lectura. Ello significa que se habrán elaborado todos los resultados parciales de la primitiva que le son necesarios al Programa Operativo para la confección de la siguiente, esto es, cumplimiento de la cualificación,

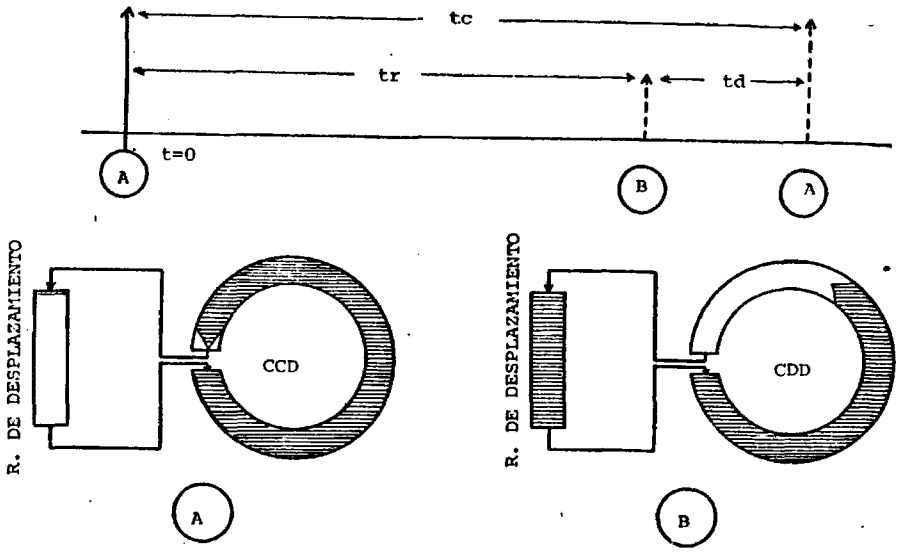


Figura III-28

funciones de agregación, etc.

Por otra parte, hasta que el primer bit de datos no se encuentre posicionado nuevamente en el punto de lectura de la CCD, no es necesario que la fase de carga de las células para el siguiente ciclo haya finalizado.

Teniendo en cuenta lo anterior, los subprocesos del Programa Operativo se distribuyen de la siguiente forma:

durante tr
- Análisis de las instrucciones actuales
- Determinación de los recursos que cada instrucción necesita
- Resolución de conflictos cuando más de una instrucción necesita un mismo recurso
- Creación y distribución de primitivas a falta de los valores de parámetros generados por las primitivas en ejecución

durante td { - Recogida de datos intermedios generados en las células
- Distribución de valores de los parámetros pendientes

REFERENCIAS

(1) FERNANDEZ DE LAS HERAS, R., "Implementación de un microprocesador no-numérico, de propósito especial, para gestión de bases de datos". Memoria de Licenciatura, Facultad de Ciencias Físicas, Univ. Complutense (Nov. 1979).

(2) PANIGRAHI, G., "Charge-Coupled Memories for Computer System". Computer, Vol. 9, n° 4, April 1976, pp.33-41.

(3) BOBECK, A.H., BOYHARD, P.I. and GEUSIC, J.E., "Magnetic Bubbles-An Emerging New Memory Technology". Proceedings IEEE, Vol. 63, n° 8, August 1975, pp. 1176-1195.

(4) LUQUE, E., RUZ, J.J., y RIPOLL, A., "Arquitectura de un microprocesador no-numérico para soportar funciones elementales de bases de datos". Actas del IV Congreso de Informática y Automática. Madrid, octubre 1979, pp. 1273-1283.

(5) DEFIORE, C.R. and BERRA, P.B., "A Data Management System Utilizing an Associative Memory", Proc. National Computer Conference, 42 (1973) pp. 181-185.

(6) LIN, S.C., SMITH, C.C.P., and SMITH, J.M., "The design of a rotating associative memory for relational Database application", ACM TODS, Vol. 1, n° 1, March 1976, pp. 53-75.

(7) LIPOVSKI, G.J., "Non-Numeric Architecture", Internal Report. Department of Electrical Engineering, University of Texas, Austin, Texas.

(8) LANGDON, G.G., "A note on Associative Processors for Data Management", ACM TODS, Vol. 3, n° 2, June 1978, pp. 148-158.

(9) OZXARAHAN, E.A., SCHUSTER, S.A. and SMITH, K.C., "RAP-Associative Processor for Data-Base Management", AFIPS Conference Proceedings, 44, 1975, pp. 379-388.

(10) RUZ ORTIZ, J.J., "Hardware method for context searching in intelligent secondary memory", IEE Computer and Digital Techniques. Vol. 2, n° 5, October 1979, pp. 223-225.

CAPITULO IV

ESTUDIO COMPARATIVO DEL RENDIMIENTO ("PERFORMANCE") DEL PROCESADOR

IV.1.- INTRODUCCION

El primer problema que se plantea ante el estudio de "performances" de cualquier sistema informático es el de la caracterización de un criterio de medida válido y significativo. Es decir, la identificación de un conjunto de variables, fácilmente medibles, que representen en forma cuantitativa:

a) la carga impuesta al sistema (workload)

b) la capacidad de servicio del mismo.

Las primeras representan la "excitación" y las segundas la "respuesta" dada por el sistema ([1]).

Una vez fijadas las variables de carga y servicio, es necesario caracterizar el comportamiento del sistema en estudio, esto es, disponer de un método que nos permita calcular los valores que toman las variables de servicio para diferentes conjuntos significativos de valores de las de carga:

S (variables de carga) = variables de servicio

Tres métodos suelen utilizarse para tal fin:

a) medida directa sobre el propio sistema

b) simulación del comportamiento del sistema

c) modelación analítica del sistema

El primero requiere la existencia física de un prototipo, lo

que no siempre es conveniente, sobre todo cuando los objetivos de la medida se encaminan a la investigación de nuevas arquitecturas. La función de carga que suele utilizarse en estos casos es de la misma naturaleza que la carga real, esto es, compuesta por programas y comandos ejecutables sobre el propio sistema. Las variables de servicio se obtienen por medida física sobre el prototipo en tiempo real.

La simulación es ampliamente utilizada en procesos de diseño. Las modificaciones introducidas en el sistema pueden ser fácilmente efectuadas y rápidamente evaluadas. Mediante un programa de ordenador, denominado simulador, se reproduce el comportamiento del sistema. Los datos de entrada al programa representan la carga utilizada; los de salida la medida de las variables de servicio.

Cuando la función de comportamiento S es expresable matemáticamente, el resultado obtenido es un modelo analítico del sistema. Resulta extremadamente difícil encontrar modelos analíticos precisos para sistemas medianamente complejos, sobre todo si se tiene en cuenta que la función de carga ha de poder representarse también en forma analítica. Sin embargo, este método es bastante útil para poner de manifiesto determinados aspectos de un sistema.

En una base de datos, las variables de servicio que más interés presentan son las que miden los tiempos de respuesta y capacidad de procesamiento (throughtput) del sistema, al flujo de accesos de llegada. Sin embargo, para nuestro caso, la caracterización de un flujo de llegada resulta bastante problemática, sobre todo si tenemos en cuenta que ello va a depender fuertemente del contexto de funcionamiento de la base, es decir, naturaleza del sistema real en ella representado, número y necesidades de los usuarios que acceden, prioridades establecidas, etc.

La alternativa seguida ha sido elegir un conjunto de accesos, representativos en la gestión de BD, y calcular sus tiempos de respuesta individuales. El estudio lo haremos en forma comparativa entre el PCBD y dos sistemas: uno convencional, constituido por un ordenador principal y su memoria secundaria; y otro procesador especialmente concebido para gestión de BDs: RAP ([2]) diseñado en la Universidad de Toronto.

Evidentemente, este estudio no pondrá de manifiesto las mejoras introducidas por la capacidad de concurrencia del PCBD. Sin embargo, sería fácil restar de los tiempos de respuesta obtenidos en funcionamiento no concurrente, la componente de tiempo en que resultaría disminuido cada acceso, cuando se conociese la medida de la diversificación de las relaciones referenciadas por un flujo de llegada concreto (factor determinante del aprovechamiento de la concurrencia en el PCBD) y considerásemos una estrategia de asignación de recursos adecuada. No obstante, veremos en este estudio que, aun para accesos individuales, las mejoras introducidas en el diseño del PCBD son bastante notorias.

La caracterización del comportamiento de los tres sistemas respecto al tiempo de respuesta (variable de servicio) invertido en la ejecución de los accesos individuales (variables de carga) la haremos analíticamente, mediante modelos con precisión de primer orden, esto es, despreciando los tiempos de CPU frente a los implicados en las transmisiones de datos.

IV.2.- CARACTERISTICAS DE LOS CRITERIOS DE SELECCION DE DATOS QUE AFECTAN A LA EFICIENCIA DE SU EVALUACION

La complejidad de las expresiones booleanas de cualificación que pueden utilizarse como criterios de búsqueda en una BD puede ser al-

tamente variable. Sin embargo, cuatro características de su estructura tienen influencia notoria respecto a la eficiencia de su evaluación:

- Número de condiciones simples (términos)
- Proporción de condiciones simples con operador igual (=)
- Número de conjunciones (disyunciones) de su forma disyuntiva (conjuntiva) normal
- Condiciones simples por contexto

El número de condiciones simples en la expresión de cualficación que denotaremos por LC (Longitud de la Cualificación), afecta al Sistema Convencional y a RAP ([3]). El PCBD puede evaluar en una sola nvolución cualificaciones con tantas condiciones simples como dominios tenga la relación referenciada.

La proporción de condiciones simples con prueba por igualdad sólo afecta al sistema convencional. Los procesadores RAP y PCBD evalúan de forma equivalente condiciones simples con cualquier operador de comparación (<,≤,=,≥,>). En este estudio distinguiremos únicamente los los casos extremos:

- Selección por igualdad. Cuando todos los términos de la cualificación contienen el operador igual (=)
- selección por desigualdad. Cuando ninguno de los términos contiene el operador igual.

El número de términos de la expresión normal de la función booleana de cualificación afecta a los tres sistemas. Para no complicar la exposición sólo consideraremos, por lo que respecta a esta característica, dos casos:

- selecciones monotérmino. Cuando la cualificación es conjuntiva

o disyuntiva simple (un solo término conjuntivo o disyuntivo)

- selecciones multitérmino. Cuando la cualificación no es simple.

Respecto a la existencia de términos por contexto en la cualificación hay que hacer notar la independencia del PCBD y el bajo rendimiento que para estas selecciones presenta el procesador RAP.

IV.3.- FORMULACION ANALITICA DE LOS TIEMPOS DE SELECCION

En este apartado estudiaremos el comportamiento funcional de cada uno de los sistemas a fin de expresar analíticamente los tiempos promedios de selección respectivos.

Para todas las variables significativas utilizaremos valores promedio, a pesar de que, en casos concretos, algunos puedan tener distribuciones con alta varianza ([4]). No hay que olvidar que el estudio lo haremos en forma comparativa y que todas las suposiciones serán idénticas para los tres sistemas.

IV.3.1.- Sistema Convencional

Para el estudio comparativo de "performance" entre el PCBD y un Sistema Convencional, soportando ambos un SGBD relacional, se presentan varias alternativas a la hora de fijar una implementación concreta para este último. Se ha elegido el método en que se crean listas invertidas para todos los dominios de cada relación. En cuanto a eficiencia, este método se considera equivalente a cualquier otro que proporcione selecciones booleanas arbitrarias ([5])([6])([7]).

Los tuples se almacenan en registros de archivos directamente

direccionables. Un registro queda únivocamente especificado con el nombre del archivo y su posición relativa dentro del mismo.

Para cada dominio de la relación se crea un archivo invertido que consta de un registro por cada valor de dominio diferente presente en la relación. Dicho registro contiene una lista de direcciones de los tuples en la relación que poseen dicho valor de dominio. Cada registro de un archivo invertido se conoce con el nombre de lista invertida. Las direcciones de tuple en una lista invertida se almacenan en orden ascendente con el fin de facilitar las operaciones de intersección y unión necesarias en la evaluación de cualificaciones.

Las listas invertidas para los dominios de cada relación se almacenan en un archivo con un directorio jerárquico multinivel. Su función es proporcionar acceso rápido a las listas invertidas para cualquier valor del dominio. Todos los niveles del directorio, salvo el más bajo, se consideran contenidos en memoria principal. Este, denominado pista índice, residirá sobre el dispositivo de memoria secundaria.

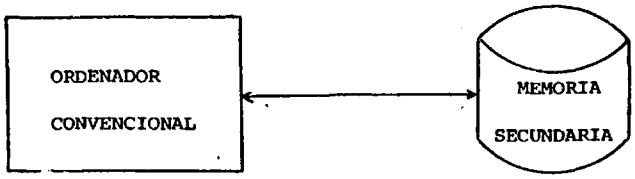
El Sistema Convencional (Figura IV-1) contendrá, pues, dos tipos de archivos para cada relación: a) un archivo índice (listas invertidas) y b) un archivo de datos (relación). Ambos residirán sobre la memoria secundaria.

Una recuperación secuencial de datos en el Sistema Convencional tardará, en promedio, un tiempo dado por ([3]):

$$RECSE(NP) = (0,5 + NP) \cdot RECON \qquad (1)$$

donde NP es el número de pistas de datos recuperados y RECON el tiempo de rotación de la memoria secundaria. Se supone que esta última es de cabezas fijas (una por pista) con señales de indicación de la posición

angular. De esta forma no es necesario la realización de rotaciones suplementariás en el acceso.



a) Sistema Convencional

	DOMINIO A	DOMINIO B	DOMINIO C	...
t1	A6	B3	C1	
t2	A1	B1	C1	
t3	A5	B1	C1	
t4	A3	B2	C2	
t5	A1	B2	C2	
t6	A1	B3	C2	
⋮	⋮	⋮	⋮	

b) Relación R

A1	t2	t5	t6	
A3	t4			
A5	t3			
A6	t1			

B1	t2	t3	
B2	t4	t5	
B3	t1	t6	

C1	t1	t2	t3	
C2	t4	t5	t6	

c) Archivos invertidos de la relación R

Figura IV-1.- Representación de datos en el sistema Convencional

Una vez conocida la dirección de un tuple, el tiempo necesario para recuperarlo (RECTU) será, en promedio, la mitad de una revolución.

$$RECTU = RECON/2 \qquad (2)$$

Para recuperar una lista invertida, el tiempo promedio (RECLI) vendrá dado por:

$$RECLI = 2 \cdot RECON/2 = RECON \qquad (3)$$

puesto que habrá que recuperar, en primer lugar, la pista índice correspondiente, y una vez procesada en la memoria principal, la lista invertida.

Para modificar una lista invertida, ésta debe ser leída, modificada en memoria principal y reescrita en la siguiente revolución. El tiempo promedio necesario para ello será:

$$MODLI = RECLI + RECON = 2 \cdot RECON \qquad (4)$$

Consideraremos a continuación la cantidad de tiempo requerida para el procesamiento de las listas invertidas a fin de evaluar una expresión de cualificación, es decir, una selección booleana. Ya que el procesamiento propiamente dicho se realiza en el ordenador, la única componente del tiempo que tendrá significación será la correspondiente a la transferencia de listas invertidas entre memoria secundaria y principal. Esto viene justificado por el hecho de que la precisión de los resultados será sólo de primer orden; estudios de mayor detalle no aportan diferencias significativas ([4]).

Si en todos los términos de la expresión de cualificación aparece el operador "=", sólo una lista invertida habrá de recuperarse. Sin embargo, si los operadores que aparecen son distintos (<,>), en promedio, la mitad de las listas deben transmitirse al ordenador para ser procesadas. Como dijimos en el apartado IV.2, tan sólo consideraremos los casos extremos (igualdad, desigualdad). Los tiempos promedios de selección boo-

leana (SELCON) en el Sistema Convencional para cada caso vendrán dados por:

$$SELCON\ (igualdad) = LC \cdot RECLI \qquad (5)$$

$$SELCON\ (desigualdad) = LC \cdot RECSE(NPAI/2) \qquad (6)$$

donde NPAI es el número de pistas ocupadas, en promedio, por un archivo invertido de un dominio de una relación.

Si por el contrario, la cualificación contiene una condición simple por contexto, a las expresiones anteriores habrá que sumarle el tiempo correspondiente a la evaluación de este término, lo cual implicará la lectura de los dos archivos invertidos completos correspondientes a los dominios que se comparan

$$SELCON\ (contexto) = SELCON\ (igualdad, desigualdad) + 2 \cdot RECSE\ (NPAI) \qquad (7)$$

IV.3.2.- Procesador RAP

Se trata de un procesador concebido en torno a la aplicación del principio de lógica distribuida sobre una memoria secundaria convencional: un sistema de discos magnéticos con dos cabezas por pista (lectura y escritura). Cada par de cabezas dispone de un elemento de procesamiento asociado, siendo el conjunto gobernado por un controlador. Su organización es del tipo SIMD (Single-Instruction, Multiple-Data), esto es, todos los elementos de procesamiento ejecutan la misma acción en paralelo distribuida por el controlador. Funciona como un back-end de un ordenador principal del que recibe las acciones (instrucciones) que el controlador distribuye. Los datos se almacenan sobre las pistas según el modelo relacional. Cada pista va encabezada por la información descriptiva del for-

mato de los tuples que contiene, los cuales pertenecen a una única relación.

Por lo que respecta a su capacidad selectiva, necesaria para este estudio, destacaremos las dos siguientes características:

1) Evalúa directamente, en una revolución, cualificaciones conjuntivas o disyuntivas simples, con tantos términos como comparadores dispongan los elementos de procesamiento. Este parámetro lo denotaremos por k.

Ello significa que el número de revoluciones necesario para evaluar una cualificación monotérmino de longitud LC será LC/K. Si la cualificación es multitérmino, en el peor de los casos, al menos dos términos de la LC serán procesados en una revolución (expresión conjuntiva (disyuntiva) con dos términos simples por disyunción (conjunción)). En este caso, el número de revoluciones necesario para su evaluación vendrá dado por CL/2 + 1.

Los tiempos promedios de selección booleana (SELRAP) en el sistema RAP para cada tipo de cualificación vendrán dados por:

$$\text{SELRAP (monotérmino)} = (0{,}5 + LC/K) \cdot RERAP \qquad (8)$$

$$\text{SELRAP (multitérmino)} = (1{,}5 + LC/2) \cdot RERAP \qquad (9)$$

siendo RERAP el tiempo de rotación del disco sobre el que se soporta el RAP.

2) Los elementos de procesamiento asociados a las pistas no tienen capacidad para evaluar términos por contexto. Ello significa que, para resolver este tipo de cualificaciones, el procesador debe invertir dos revoluciones por tuple existente en la relación: una para recuperar uno de los dominios implicados en la comparación y, otra para compararlo -utilizándolo como constante de un término por contenido- con el otro dominio del mismo tuple ([8]).

El hecho de ser este número igual al de tuples en una relación y no en una célula (elemento de procesamiento-pista) es consecuencia directa de la organización SIMD del RAP. No obstante, para nuestro estudio comparativo, consideraremos el último caso, puesto que ello sólo implicaría la inicialización individual de las células componentes de la relación.

Designando a dicho número por NTC, el tiempo promedio de selección en el RAP bajo cualificaciones que contienen un término por contexto vendrá dado por:

$$SELRAP\ (contexto) = SELRAP\ (monotérmino,\ multitérmino) + 2\ .\ NTC \qquad (10)$$

IV.3.3.- Procesador PCBD

Respecto a la capacidad selectiva del PCBD hemos de señalar:

1) Resuelve directamente, en una revolución, cualificaciones conjuntivas o disyuntivas simples con tantos términos como dominios tenga la relación referenciada. Por tanto, el número de revoluciones para seleccionar con cualificaciones monotérmino será en el PCBD 1, independiente de LC. En el caso de los multitérmino ocurrirá igual que en el RAP, esto es, en el caso más desfavorable, el número de revoluciones será $LC/2 + 1$.

El tiempo promedio de selección booleana (SELPCBD) será, pues:

$$SELPCBD\ (monotérmino) = 1{,}5\ .\ REPCBD \qquad (11)$$

$$SELPCBD\ (multitérmino) = (1{,}5 + LC/2)\ .\ REPCBD \qquad (12)$$

siendo REPCBD el tiempo de ciclo PCBD (= tc).

2) La presencia de términos por contexto no afecta al tiempo de selección, por tanto:

SELPCBD (contexto) = SELPCBD (contenido)

IV.4.- DATOS COMPARATIVOS ENTRE LOS TRES SISTEMAS

A fin de comparar en forma cuantitativa los diferentes tiempos promedios invertidos en los accesos, es necesario fijar un tamaño promedio para una relación y ver su repercusión sobre los parámetros respectivos en los tres sistemas.

Si fijamos en el RAP y el PCBD el número de células que soportan esta relación promedio a NC = 100, con una capacidad para los elementos de memoria de cada célula de 1/2 Mbits, resulta según (3) los siguientes valores:

- número promedio de pistas ocupadas por un archivo invertido de un dominio (atributo) de esta relación en el Sistema Convencional:

$$NPAI = 50 \qquad (13)$$

- relación del tiempo de revolución entre el RAP y el Sistema Convencional:

$$RERAP = 2{,}5 \cdot RECON \qquad (14)$$

Si tomamos como valor promedio para la longitud de un tuple de 1/2 kbits, el número promedio de tuples por célula será:

$$NTC = 1.000 \qquad (15)$$

Fijando para la memoria CCD del PCBD una razón de bits 4 veces superior a la de los discos magnéticos ([9]) y teniendo en cuenta (14), resultan los siguientes valores para las razones de revolución:

$$RECON/REPCBD = 1{,}6 \qquad (16)$$

$$RERAP/REPCBD = 4 \qquad (17)$$

Según (3) fijaremos el número K de comparadores por célula en RAP a:

$$K = 5 \qquad (18)$$

La operación de salida de datos, previamente seleccionados, aportará una componente al tiempo total invertido en la realización de los diferentes tipos de recuperación que estudiaremos en los siguientes apartados. Será necesario, pues, cuantificar su valor para cada uno de los sistemas.

En el Convencional, conocido el número de datos (tuples) seleccionados ND, su valor vendrá dado por (1).

Para los procesadores RAP y PCBD los datos seleccionados en las diferentes células se recuperan vía un único bus. Por tanto, el número de revoluciones NR necesario para recuperar ND tuples seleccionados, será igual al número máximo de estos tuples coincidentes en la misma posición angular de sus respectivos elementos de memoria. Este número, desconocido a priori, dependerá en general de:

- número de células ocupadas por la relación (NC)
- número de posiciones de tuple por célula (NTC)
- número de datos seleccionados ND y forma de distribución entre las NTC posiciones de las NC células.

Asumiendo una distribución uniforme para ND entre células y posiciones, se puede evaluar la función NR(ND) utilizando el método de simulación de Monte Carlo ([10])([11]). En la Figura IV-2 hemos representado gráficamente el resultado para los valores anteriormente fijados NC=100, NTC=1.000.

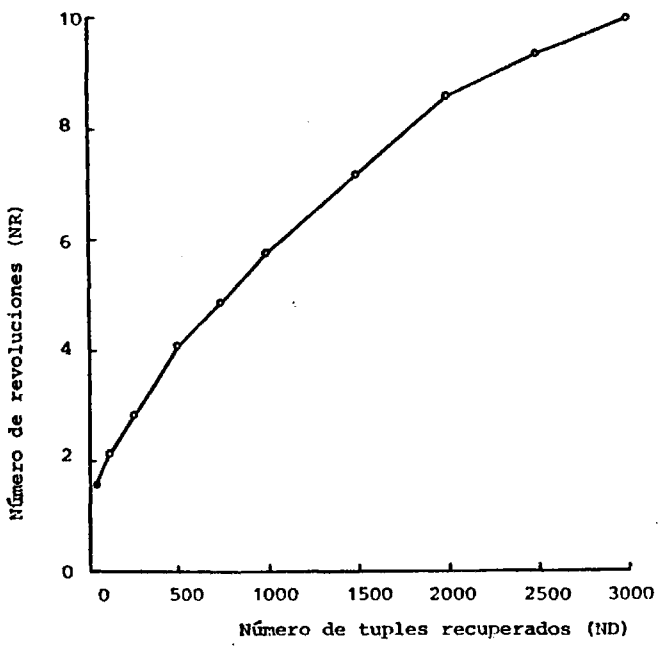


Figura IV-2.- Representación gráfica de la función NR(ND) evaluada por simulación de Monte Carlo

IV.5.- RECUPERACION BOOLEANA SOBRE UNA UNICA RELACION

En este apartado compararemos los tiempos de respuesta promedios del PCBD frente a los procesadores RAP y CONVENCIONAL, respecto a recuperaciones bajo un criterio de búsqueda booleano, es decir, no incluyendo funciones de agregación. Además, todos los términos de la expresión booleana referenciarán dominios de una única relación.

En los tres sistemas, en general, el tiempo de recuperación tendrá dos componentes: a) la necesaria para la selección de datos que cum-

plen la expresión booleana y b) la invertida en la recuperación física de datos ya seleccionados. Sin embargo, en el PCBD la instrucción RE selecciona datos al tiempo que los transmite a la U.C. Ello significa que ambos procesos no son completamente secuenciales.

Dependiendo de la existencia o no, en la expresión booleana de cualificación, de términos que implican la comparación entre sí de dos dominios de un mismo tuple, distinguiremos, en los dos subapartados siguientes, las recuperaciones por contenido y por contexto.

IV.5.1.- Recuperación booleana por contenido

Estudiaremos, en primer lugar, la expresión analítica del tiempo de respuesta que, en cada uno de los tres sistemas, requieren este tipo de recuperaciones:

- CONVENCIONAL

Si después del tiempo SELCON necesario para seleccionar los registros que cumplen la cualificación resultan ser un número ND, el tiempo promedio de recuperación en este sistema (RECCON) vendrá dado por:

$$RECCON = SELCON + ND \cdot RECTU$$

siendo RECTU el tiempo promedio de recuperación de un registro (tuple) una vez conocida su dirección, esto es, una vez seleccionado. Teniendo en cuenta (5), (2), (6), (3), (1) y (13), tendremos la siguiente expresión para RECCON en los dos tipos de selección contemplados en el apartado IV.2:

$$RECCON = \begin{cases} (ND/2 + LC) \cdot RECON & \text{(igualdad)} \\ (ND/2 + 25{,}5 \cdot LC) \cdot RECON & \text{(desigualdad)} \end{cases}$$

- Procesador RAP

Teniendo en cuenta que en este procesador la selección no se solapa en el tiempo con la recuperación física y que esta última exige un número de revoluciones NR(ND), evaluadas por simulación de Monte Carlo, el tiempo promedio de recuperación (RECRAP) vendrá dado por:

$$RECRAP = SELRAP + NR(ND) \cdot RERAP$$

Teniendo en cuenta (8) y (9) para selecciones monotérmino y multitérmino respectivamente, resulta:

$$RECRAP = \begin{cases} (0,5 + LC/5 + NR(ND)) \cdot RERAP & \text{(monotérmino)} \\ (1,5 + LC/2 + NR(ND)) \cdot RERAP & \text{(multitérmino)} \end{cases}$$

- Procesador PCBD

En el PCBD, las cualificaciones monotérmino (conjuntivas o disyuntivas simples) se resuelven en paralelo con la transmisión de tuples a la U.C. El tiempo promedio de respuesta contendrá únicamente la componente de recuperación de datos seleccionados (11). Para las multitérmino, tan sólo existirá solapamiento en la evaluación de la última componente conjuntiva o disyuntiva de su expresión normal, esto es, en una revolución. La expresión analítica será pues:

$$REC\ PCBD = \begin{cases} NR(ND) \cdot REPCBD & \text{(monotérmino)} \\ SELPCBD + (NR(ND) - 1) \cdot REPCBD & \text{(multitérmino)} \end{cases}$$

Teniendo en cuenta (12), resulta:

$$REC\ PCBD = \begin{cases} NR(ND) \cdot REPCBD & \text{(monotérmino)} \\ (1,5 + LC/2 + NR(ND)) \cdot REPCBD & \text{(multitérmino)} \end{cases}$$

Para unificar criterios de comparación entre el PCBD y el Sistema Convencional, sólo consideraremos los siguientes tipos extremos de expresiones booleanas de recuperación por contenido:

- simples: por igualdad y monotérmino

- complejas: por desigualdad y multitérmino

En la Figura IV-3 hemos representado en forma gráfica la razón de recuperación:

$$\begin{cases} \dfrac{\text{RECCON (igualdad)}}{\text{RECPCBD (monotérmino)}} = \dfrac{(LC + ND/2)\ .\ RECON}{NR(ND)\ .\ REPCBD} & \text{(simple)} \\ \dfrac{\text{RECCON (desigualdad)}}{\text{RECPCBD (multitérmino)}} = \dfrac{(25{,}5\ .\ LC + ND/2)\ .\ RECON}{(1{,}5 + LC/2 + NR(ND))\ REPCBD} & \text{(compleja)} \end{cases} \quad (16)$$

para varios valores de LC y razón de revoluciones RECON/REPCBD dada por

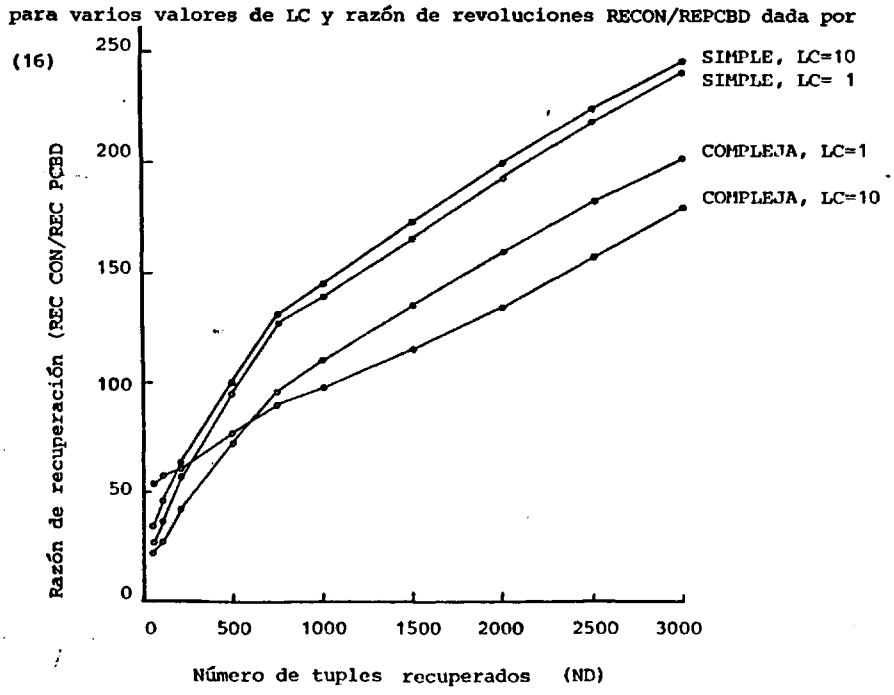


Figura IV- 3.- Recuperación booleana por contenido (CONVENCIONAL-PCBD)

Puede observarse el carácter creciente de la razón, casi en proporción directa con el número de datos recuperados. En buena medida ello es consecuencia de que la proporción, en el sistema convencional, de datos

transmitidos entre memoria secundaria y ordenador y datos útiles recuperados en el acceso, va aumentando a medida que aumentan estos últimos como consecuencia del procesamiento de las listas invertidas. La presencia de expresiones booleanas multitérmino no favorecen el mejor rendimiento del PCBD.

Análogamente, en la Figura IV-4 hemos representado la razón de recuperación para los procesadores RAP y PCBD:

$$\frac{\text{REC RAP}}{\text{RECPCBD}} = \begin{cases} \dfrac{(0{,}5 + LC/5 + NR(ND)) \cdot RERAP}{NR(ND) \cdot REPCBD} & \text{(monotérmino)} \\ \dfrac{RERAP}{REPCBD} & \text{(multitérmino)} \end{cases}$$

con LC como parámetro y razón de revolución dada por (17)

Para las recuperaciones con expresiones multitérmino, la mejora presentada por el PCBD frente al RAP deriva exclusivamente de razones tecnológicas: la mayor razón de transferencia de bits de las memorias CCD frente a los discos magnéticos. En cambio, cuando la expresión es monotérmino, la razón de recuperación se hace creciente: consecuencia de la capacidad del PCBD para solapar la selección de tuples con su recuperación física. El favorable aumento de la razón con LC deriva de la posibilidad del PCBD para evaluar directamente cualificaciones conjuntivas o disyuntivas simples con tantos términos como dominios tenga la relación referenciada, al tiempo que en el RAP la evaluación directa está limitada por el número de comparadores por célula.

IV.5.2.- Recuperación booleana por contexto

Cuando la expresión booleana de recuperación contiene términos que implican la comparación de dos dominios de un mismo tuple, las ventajas PCBD resultan notorias, especialmente frente al procesador RAP.

Para no complicar excesivamente el análisis, consideraremos la presencia de un solo término por contexto en la expresión de cualificación:

- CONVENCIONAL

Los términos por contexto sólo tienen influencia sobre el proceso de selección. Por tanto, la expresión del tiempo promedio de recuperación para este tipo de expresiones vendrá dada por (8) sustituyendo SELCON por (7), esto es:

$$RECCON = \begin{cases} (ND/2 + LC + 2 \cdot RECSE(NPAI)) \cdot RECON & \text{(igualdad)} \\ (ND/2 + 25{,}5 \cdot LC + 2 \cdot RECSE(NPAI)) \cdot RECON & \text{(desigualdad)} \end{cases}$$

Teniendo en cuenta (13) y (1), resulta:

$$RECCON = \begin{cases} (ND/2 + LC + 101) \cdot RECON & \text{(igualdad)} \\ (ND/2 + 25{,}5 \cdot LC + 101) \cdot RECON & \text{(desigualdad)} \end{cases}$$

- Procesador RAP

Igual que en el caso anterior el término por contexto sólo aumenta la componente de selección. Teniendo en cuenta (10) y (15), tendremos:

$$RECRAP = \begin{cases} (2000{,}5 + LC/5 + NR(ND)) \cdot RERAP & \text{(monotérmino)} \\ (2001{,}5 + LC/2 + NR(ND)) \cdot RERAP & \text{(multitérmino)} \end{cases}$$

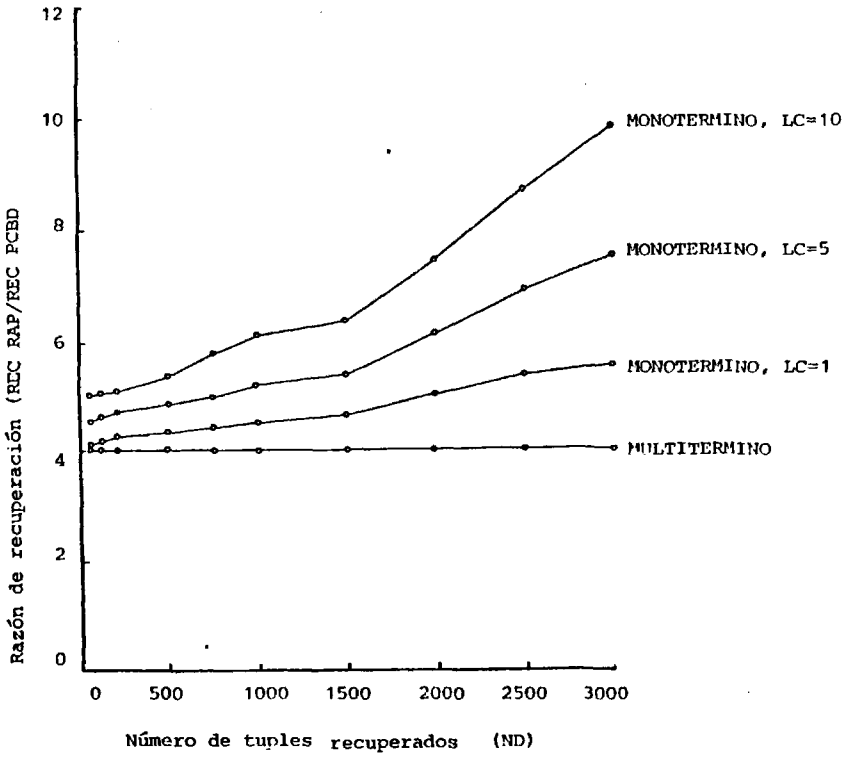


Figura IV- 4.- Recuperación booleana por contenido (RAP-PCBD)

- Procesador PCBD

El término por contexto no introduce ninguna componente de tiempo respecto a la recuperación por contenido:

RECPCBD (contexto) = RECPCBD (contenido)

En la Figura IV-5 aparece la representación gráfica de la razón:

$$\begin{cases} \dfrac{\text{RECCON (igualdad)}}{\text{RECPCBD (monotérmino}} = \dfrac{(LC + ND/2 + 101)\ .\ RECON}{NR(ND)\ .\ REPCBD} & \text{(simple)} \\ \dfrac{\text{RECCON (desigualdad)}}{\text{RECPCBD (multitérmino)}} = \dfrac{(25{,}5\ .\ LC+ND/2+101).RECON}{(1{,}5+LC/2+NR(ND)).REPCBD} & \text{(compleja)} \end{cases}$$

con LC como parámetro.

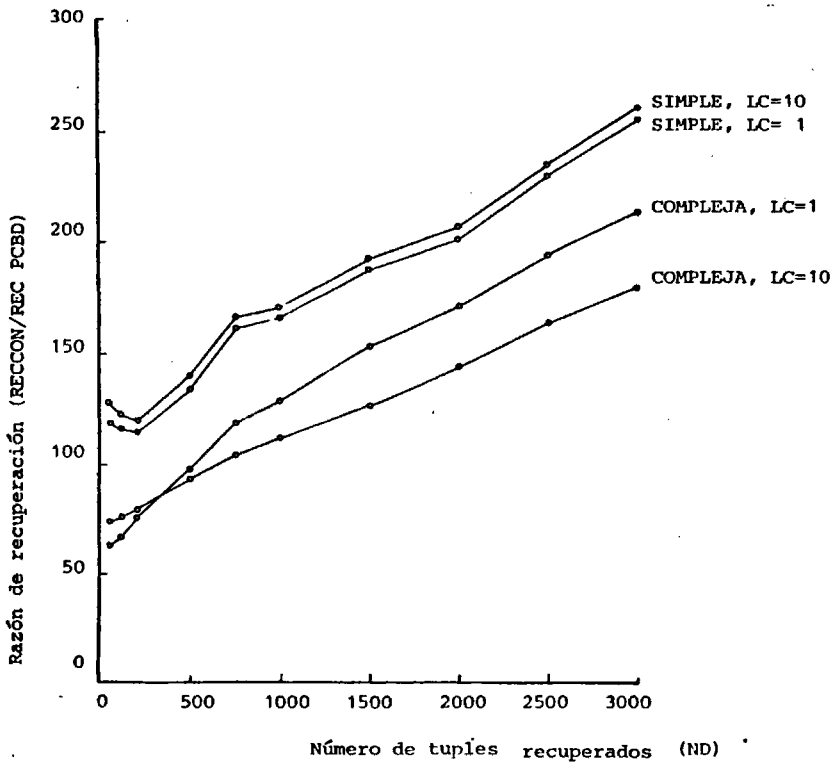


Figura IV-5.- Recuperación booleana por contexto (CONVENCIONA-PCBD)

Para las cualificaciones complejas el crecimiento de la razón es prácticamente análogo al presentado en ausencia del término por contexto

(Figura IV-3) ya que el tiempo de transmisión de las listas invertidas correspondientes a los dos dominios que se comparan representa un porcentaje pequeño respecto del que ya exige la transmisión del elevado número de listas en las expresiones con términos por desigualdad.

En cuanto a las cualificaciones simples, el aumento de la razón se hace más sensible cuando los datos seleccionados (ND) son pocos. Ello es lógico si tenemos en cuenta que la resolución de un término por igualdad exige, en el Sistema Convencional, la transmisión de una única lista invertida.

Para los procesadores RAP y PCBD la razón de recuperación por contexto:

$$\frac{RECRAP}{RECPCBD} = \begin{cases} \dfrac{(2000,5+LC/5+NR(ND)).RERAP}{NR(ND)\ .\ REPCBD} & \text{(monotérmino)} \\ \dfrac{(2001,5+LC/2+NR(ND)).RERAP}{(1,5+LC/2+NR(ND)).REPCBD} & \text{(multitérmino)} \end{cases}$$

aparece representada gráficamente en la Figura IV-6.

En primer lugar hay que hacer notar el elevado valor de la razón cualquiera que sea la longitud (LC) y tipo (monotérmino o multitérmino) de las cualificaciones: consecuencia directa de la ineficacia del RAP -puesto anteriormente de manifiesto- para evaluar cualificaciones por contexto.

Por lo que al crecimiento de la razón respecta, hay que significar dos hechos: a) el aumento progresivo con ND de las cualificaciones monotérmino y b) la tendencia de las multitérmino a amortiguar el crecimiento a medida que aumenta su longitud LC. Ambos hechos son el resultado de las características del PCBD puestas de manifiesto en las cualifi-

caciones por contenido.

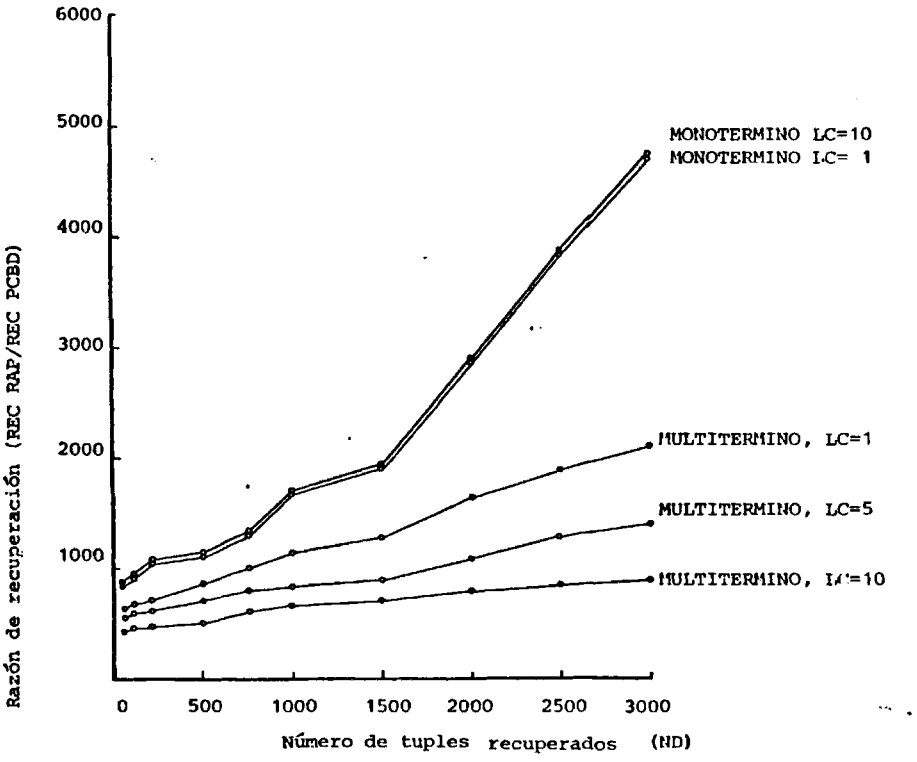


Figura IV.-6. Recuperación booleana por contexto (RAP-PCBD)

IV.6.- RECUPERACION BOOLEANA CRUZADA SOBRE DOS RELACIONES

En este tipo de accesos se utilizan los datos seleccionados en una relación, bajo determinado criterio de búsqueda, para seleccionar tuples de una segunda relación. Estos serán los que cumplan la cualificación disyuntiva sobre uno solo de sus dominios, formada con los valores de dominio de los tuples seleccionados en la primera. Ambos dominios de-

berán ser, evidentemente, compatibles.

Así, sobre la base relacional definida en el apartado I.3, el acceso: "Recuperar los números de todos los vuelos que realizan los pilotos residentes en Sevilla", será de este tipo. Su realización exigirá los siguientes pasos:

1) Seleccionar en la relación PILOTO todos los tuples que cumplen CR = SEVILLA
2) Seleccionar en la relación VUELO todos los tuples cuyo dominio PI# coincida con alguno de los valores de PI# pertenecientes a tuples seleccionados en PILOTO en el paso 1)
3) Recuperar los vuelos de VU# de todos los tuples de VUELO seleccionados en el paso 2).

Se trata, pues, de una θ-composición implícita, esto es, en la que no es necesario la formación física de la relación resultante. Evidentemente, este tipo de recuperaciones pueden extenderse a más de dos relaciones. No obstante, en este estudio sólo consideraremos el primer caso.

Veamos a continuación la expresión analítica del tiempo promedio de recuperación cruzada para cada uno de los sistemas.

- CONVENCIONAL

Aquí será necesario realizar los siguientes pasos ([3]):

a) Proceso de selección sobre la primera relación con el criterio de búsqueda especificado.

b) Recuperación secuencial de las listas invertidas para el dominio de composición de la primera relación.

c) Realizar en memoria principal la intersección de los conjuntos obtenidos en a) y b).

d) Para cada valor obtenido en c) recuperar la lista invertida para el dominio de composición de la segunda relación.

e) Realizar en memoria principal la unión de las listas obtenidas en d).

f) Para cada valor de las listas resultantes en e) recuperar el tuple de la segunda relación.

El tiempo de realización de los pasos c) y e), al ser una expresión analítica con precisión de primer orden la que buscamos, será despreciable.

El paso a) exigirá un tiempo dependiente del criterio de selección utilizado sobre la primera relación (SELCON).

Los pasos b) y d) requerirán, cada uno de ellos, el tiempo promedio de recuperación de un archivo invertido para un dominio de una relación (RECSE(NPAI)).

En el paso f) se invertirá un tiempo dependiente del número ND de tuples finalmente seleccionados (ND* RECTU).

Por tanto, la expresión del tiempo promedio de recuperación cruzada para el sistema convencional será:

$$RECRCON = SELCON + 2 \cdot RECSE(NPAI) + ND \cdot RECTU$$

- Procesador RAP

En este procesador sólo serán necesarios los pasos 1), 2) y 3) descritos anteriormente. Los tiempos promedios de los pasos 1) y 3) serán, respectivamente, los ya conocidos: SELRAP y NR(ND). El paso 2) lo realiza este procesador en un número promedio de revoluciones dado por ([3]): 1+NS/K+NC; siendo NS el número de tuples inicialmente seleccionados en el paso 1), K el número de comparadores por célula y NC el número de células que soportan la relación. Teniendo en cuenta (18) y que

NC = 100, resulta la siguiente expresión para el tiempo promedio de recuperación cruzada en RAP:

RECRRAP = SELRAP + (NS/5 + 101) . RERAP + NR(ND) . RERAP

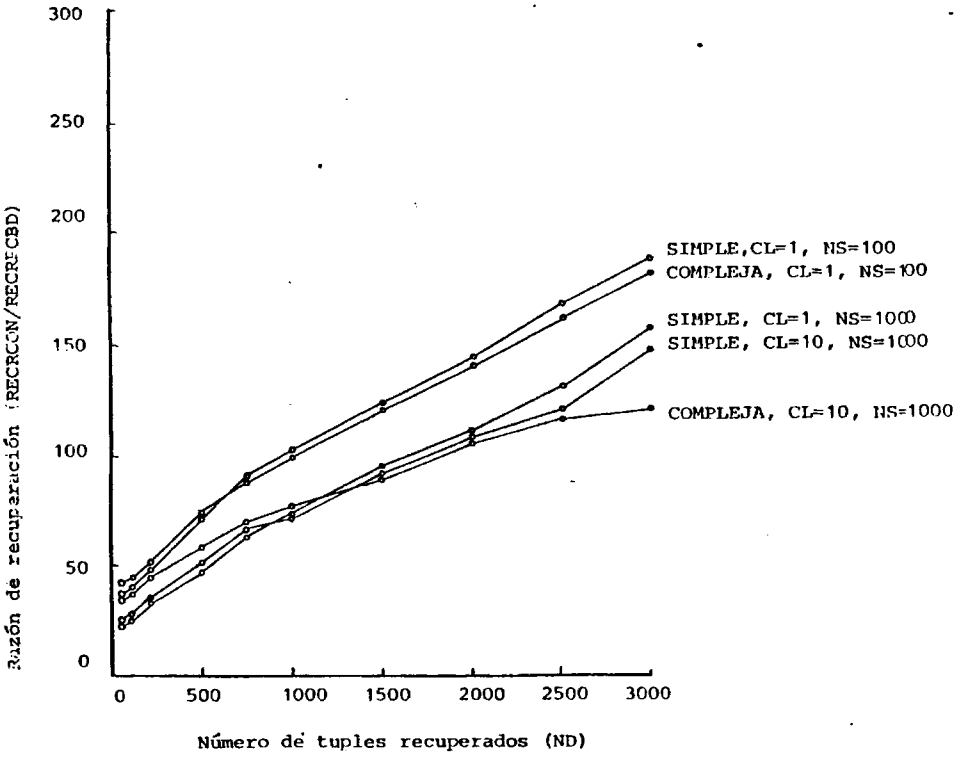


Figura IV-7.- Recuperación booleana cruzada por contenido (CONVENCIONAL-PCBD)

- Procesador PCBD

En el PCBD este tipo de recuperaciones también se resuelven con los pasos 1), 2) y 3). Los 1) y 2) se ejecutan con la instrucción de Selección Indirecta (SI), y el 3) con la de Recuperación Externa. En la realización del paso 2) interviene, pues, el mecanismo de transmisión directa

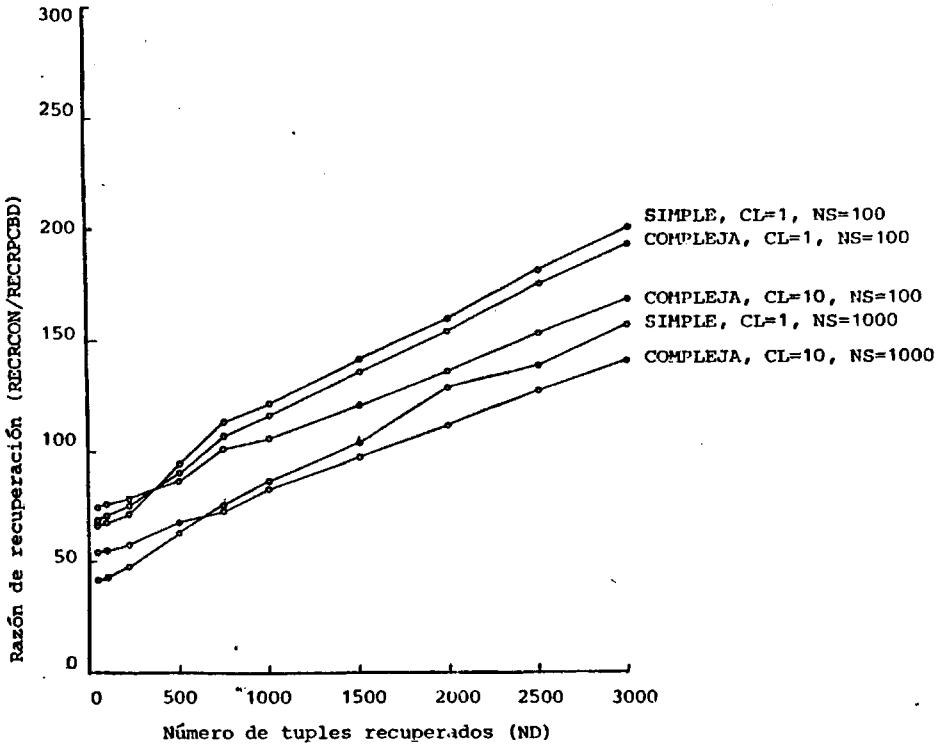


Figura IV-8.- Recuperación booleana cruzada por contexto (CONVENCIONAL-PCBD)

de datos con utilización del bus de intercambio. Ello significa que el número de revoluciones vendrá determinado por el número máximo de tuples seleccionados en el paso 1) y coincidentes en idénticas posiciones angulares, es decir, dada por una función idéntica a la evaluada para salida: NR(NS).

El tiempo de recuperación cruzada para el PCBD vendrá dado por:

RECRPCBD = SELPCBD + NR(NS) . REPCBD + NR(ND) . REPCBD

En las Figuras IV-7 y IV-8 se representa gráficamente la razón

de recuperación RECRCON/RECRPCBD para selecciones booleanas sobre la primera relación por contenido y contexto, respectivamente, y tomando como parámetros LC, NS y el tipo de cualificación (simple, compleja). En ellas se pone de manifiesto la poca influencia que la existencia de un término por contexto tiene frente al proceso de selección cruzada propiamente dicho. También puede observarse el carácter creciente de la razón con el número de datos finalmente recuperados (ND) y el decreciente con el de datos seleccionados sobre la primera relación (NS).

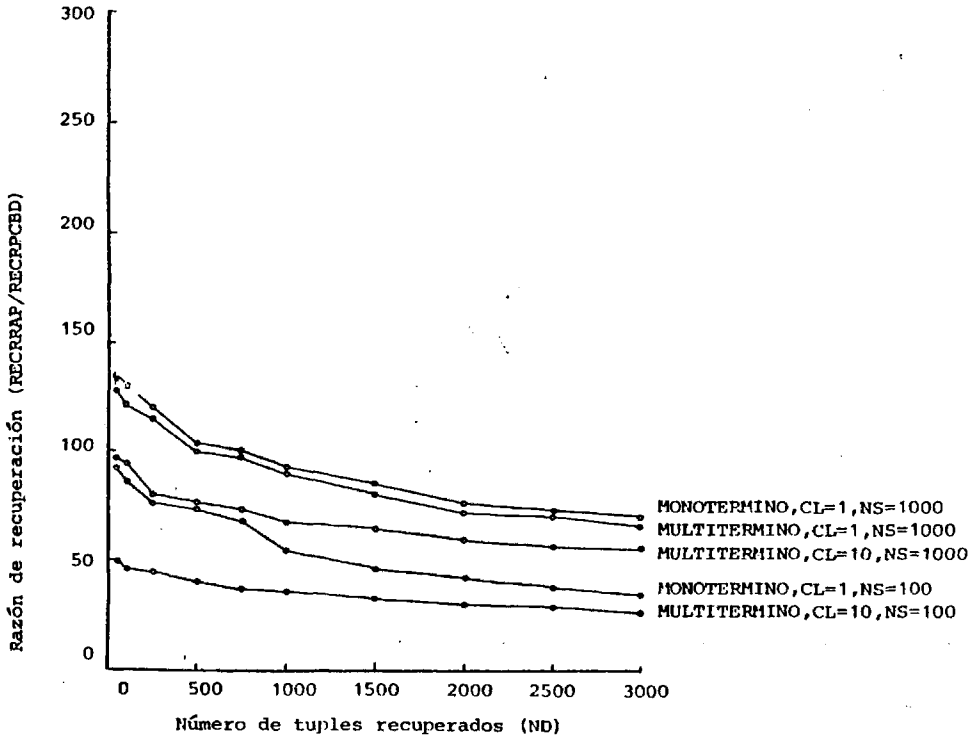


Figura IV-9.- Recuperación booleana cruzada por contenido (RAP-PCBD)

En las Figuras IV-9 y IV-10 se muestran los correspondientes a los procesadores RAP y PCBD. A pesar de su valor absoluto favorable para el PCBD éste va decreciendo a medida que el número de datos recuperados (ND) crece. Sin embargo, con el parámetro NS el comportamiento es inverso.

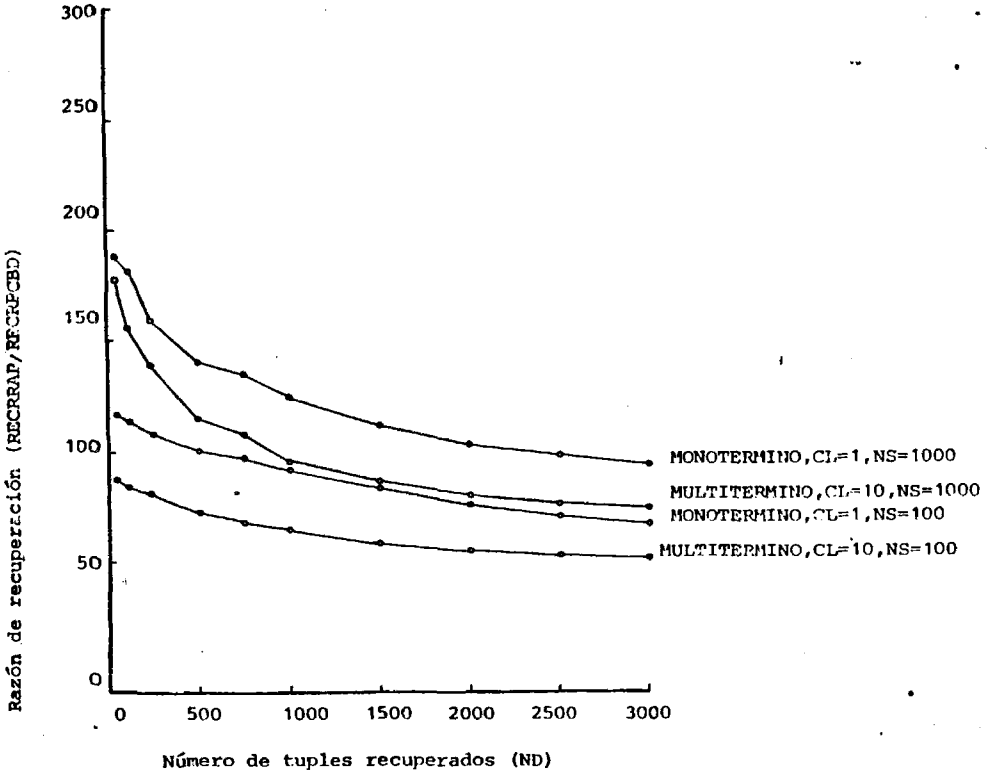


Figura IV-10.- Recuperación booleana cruzada por contexto (RAP-PCBD)

IV-7.- ACCESOS DE MODIFICACION

Estas operaciones constan de la selección de un conjunto de tuples con una expresión de cualificación y la modificación de sus dominios mediante operaciones aritméticas de suma y resta, así como sustituciones.

En el sistema convencional, después que son seleccionados en un tiempo promedio SELCON, cada uno de los ND tuples han de ser leídos, modificados y reescritos en un tiempo promedio ND . (RECTU + RECON). Finalmente, por cada dominio modificado será necesaria la actualización

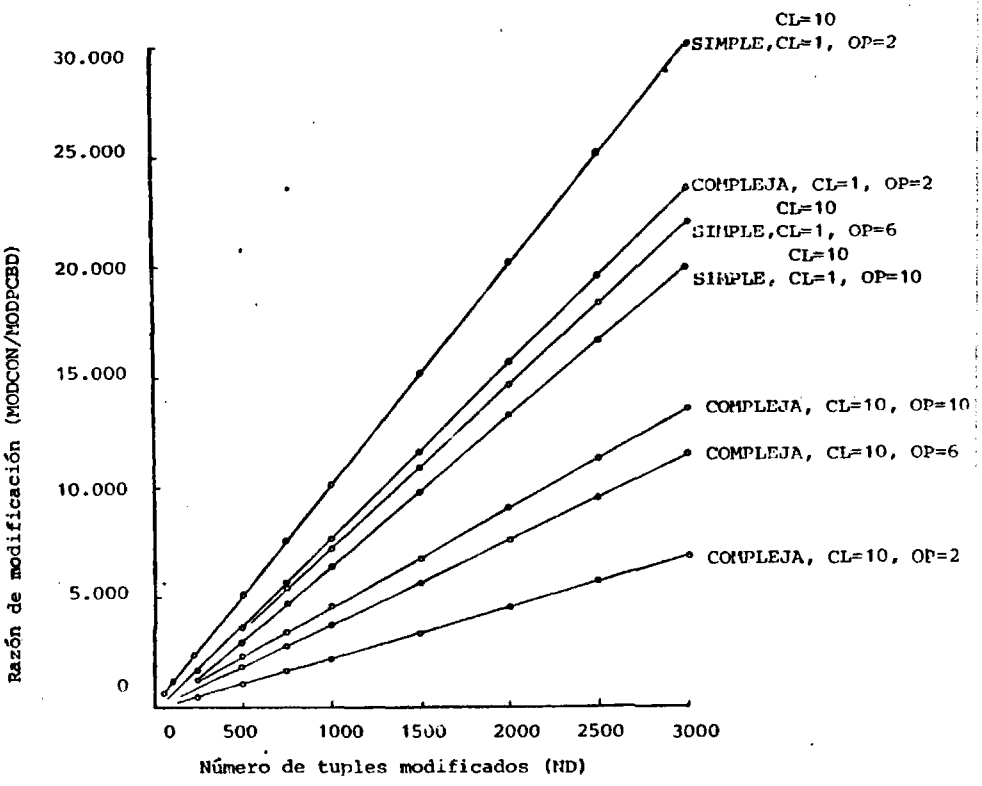


Figura IV-11.- Modificación booleana por contenido

de dos listas invertidas, esto es, un tiempo promedio: ND . 2 . MODLI. Por tanto, si es OP el número de operandos (dominios modificados) aritméticos en la expresión de modificación, el tiempo promedio necesario para este tipo de accesos en el sistema convencional vendrá dado por:

$$MODCON = SELCON + ND \cdot (RECTU + RECON) + 2 \cdot OP \cdot ND \cdot MOLI$$

En los procesadores RAP y PCBD la selección y modificación simple (una sola operación aritmética) se realizan simultáneamente, en una revolución. Sólo si la expresión de modificación contiene más de un operador aritmético, serán necesarias revoluciones adicionales, una por cada nuevo operador.

La diferencia de tiempo existente entre RAP y PCBD en este tipo de accesos sólo será de orden cuantitativo, ocasionada por la mayor razón de bits en las memorias CCD que en los discos magnéticos. Por ello, en la Figura IV-11 sólo hemos representado gráficamente la razón de modificación MODCON/MODPCBD, teniendo en cuenta que, según lo anterior, la expresión analítica del tiempo promedio de recuperación en el procesador PCBD será:

$$MODPCBD = SELPCBD + (OP-2) \cdot REPCBD$$

REFERENCIAS

(1) SHEMER, J.E., "Computer System Instrumentation and Performance Measurement". Computer, July-August 1972.

(2) OZKARAHAN, E.A., SCHUSTER, S.A., and SMITH, K.C., "RAP-An associative processor for data base management". Proc. AFIPS 1975, NCC, Vol. 44, pp. 379-387.

(3) OZKARAHAN, E.A., SCHUSTER, S.A. and SMITH, K.C., "Performance Evaluation of RAP". ACM TODS, Vol. 2, n° 2, June 1977, pp. 175-195.

(4) OZKARAHAN, E.A., SCHUSTER, S.A., and SMITH, K.C., "RAP-A data base processor". Tech. Rep. CSPG-43, Comptr. Syst. Res. Group, U. of Toronto, Toronto Canada, Sept. 1974.

(5) ASTRAHAN, M.M. and CHAMBERLIN, D.D., "Implementation of a structured English query language", Comm. ACM. Vol. 18 n° 10 (Oct. 1975), pp. 580-587.

(6) CARDENAS, A.F., "Analysis and performance of inverted data base structures", Comm. ACM, Vol. 18, n° 5 (May 1975) pp. 253-263.

(7) DATE, C.J., "An Introduction to Data Base Systems", Addison-Wesley, Reading, Mass., 1975.

(8) OZKARAHAN, E.A. and SCHUSTER, S.A., "A High-level Machine-Oriented Assembler Language for a Data Base Machine", Tech. Rep. CSRG-74. Comptr. Syst. Res. Group, U. of Toronto, Toronto, Canada, Oct. 1976.

(9) HSIAO, D.K., MADNICK, S.E., "Database Machine Architecture in the Context of Information Technology Evolution", in Proc. 3rd Int. Conf. on Very Large Data Bases, ACM, Tokyo, October 1977, pp. 63-84.

(10) METROPOLIS, N. and ULAN, S., "The Monte Carlo Method, 'Journal of American Statistics Association", Vol. 44, 1949, pp. 335-41.

(11) ROMERO, C., "Técnicas de programación y control de proyecto". Ediciones Pirámide, S.A., Madrid, 1979.

CONCLUSIONES Y PRINCIPALES APORTACIONES

1) Se realiza un estudio crítico de las diferentes alternativas actualmente en uso para modelar un sistema real sobre una base de datos, así como de las formas respectivas de acceder a ella.

2) Se estudian las líneas generales que han inspirado las diferentes máquinas propuestas para bases de datos y se pone de manifiesto la conveniencia de utilizar, junto al principio de lógica distribuida, la gestión particionada del conjunto de células y la adaptación de la arquitectura de éstas al carácter serie rotante de las memorias actualmente utilizables.

3) Se propone la organización (MIMD) de un Procesador para gestión de bases de datos con capacidad para ejecutar concurrentemente un conjunto de accesos.

4) Se representa directamente el modelo relacional sobre la estructura física de la memoria, eliminando todo tipo de información asociada.

5) Se consigue la "completitud" relacional del repertorio con tan sólo cuatro instrucciones básicas de recuperación. Además, se incorporan instrucciones para mejorar el rendimiento y facilitar la traducción de lenguajes relacionales de alto nivel.

6) Se introduce un dispositivo de intercambio directo de datos entre las células componentes del Procesador a fin de acelerar las operaciones de tipo n^2, particularmente la θ-composición implícita.

7) Se describe la realización física detallada de cada una de las unidades de las células componentes del Procesador, haciendo resaltar su capacidad selectiva de datos. Esta se traduce en la posibilidad de acceso directo por contexto, esto es, la implementación hardware de la θ-restric-

ción relacional y evaluación de cualificaciones que referencian todos los dominios de una relación en una sola revolución de la memoria.

8) Se realiza un estudio comparativo de rendimiento, donde se pone de manifiesto la repercusión favorable que, respecto a los tiempos de respuesta a accesos individuales, presenta el Procesador propuesto frente a un sistema convencional y otro procesador específico para bases de datos.

Madrid junio 1980

Reunido el Tribunal que suscribe en el día de la fecha acordó calificar la presente Tesis Doctoral con la censura de -
Sobresaliente "cum laude"
Madrid 20 de Junio 1980